- The goal is to provide a memory system with cost almost _____ level of memory and speed almost _____ level. The levels of the hierarchy usually subset one another. All data in one level is also found in the level below, and all data in that lower level is found in the one below it, and so on until we reach the bottom of the hierarchy.

A. as low as the cheapest \ as large as the fastest

B. as low as the fastest \ as fast as the cheapest

C. as low as the cheapest \ as fast as the fastest

D. as low as the fastest \ as low as the cheapest

- Which of the following statements about two-level cache is NOT correct?

  A. With two-Level cache, we can decrease miss penalty.

  B. The first-level cache should be large enough to obtain a small miss rate.

  C. The second-level cache generally uses a bigger block size than that of the first-level cache.

  D. The second-level cache should be large enough and use higher association to catch almost all memory accesses in the second level.

- Which of the following views is correct?

  A. Since the focus is shifted from computation to communication and storage of information, the importance of I/O is increasing fast than ever.

  B. The I/O performance doesn't matter because the processor is so fast that it always need to wait for human' feedback.

  C. The I/O performance doesn't matter because the CPU will run another task when a process waits for a peripheral, the throughput does not descend.

- Which RAID level waste most of the storage?

A. RAID 1

B. RAID 0

C. RAID3

D. RAID 5

- RAID (Redundant Array of Independent Disks) is a storage technology, but _____ has no redundency.
    - A. RAID 0
    - B. RAID 1
    - C. RAID 3
    - D. RAID 5

- According to the rate of block conflict, the descend order of cache mapping techniques is _____.
    - A. fully associative、direct mapped、set Associative.
    - B. set associative、direct mapped、fully associative.
    - C. direct mapped、set associative、fully associative.
    - D. fully associative、set associative、direct mapped

- As processes working, not all objects referenced by a program need to reside in main memory. If the computer has _____, then some objects may reside on___. The address space is usually broken into fixed-size blocks, called___. At any time, each ___resides either in main memory or on .

A. cache memory \ main memory \ blocks \ block \ main memory

B. cache memory \ disk \ blocks \ block \ disk

C. virtual memory \ main memory \ pages \ page \ main memory

D. virtual memory \ disk \ pages \ page \ disk

- Which of the following policy will NOT improve virtual memory performance?

A. Write through

B. full-associative map

C. TLB cache

D. LRU replacement

- Which of the following statements about the causes of cache miss is correct?

A. The obvious way to reduce capacity misses is to increase capacity of the cache, while at the risk of longer hit time and higher cost.

B. The larger the block size isa, the better to decrease the conflict misses, because larger size take better advantage of spatial locality.

C. Use larger block size can increase compulsory misses.

D. Higher associativity can be used to reduce conflict misses, and at the same time it decrease the average memory access time too.

- In a cache-memory hierarchy system, assume that the memory size is 256MB, with a 4KB write back cache in 2-way associative. The block size is 32B. Then the size of index field of physical memory address is

  A. 5 bit

  B. 6 bit

  C. 7 bit

  D. 11 bit

  E. 17 bit

- Replacing a single memory bank with multiple interleaved memory banks has which of the following effects?

  - A. Increase latency.
  - B. Increase bandwidth.
  - C. Increase reliability.
  - D. Decrease latency

- Which one of the following expressions is the CPU TIME performance equation ?
  - A. Hit Time + Miss Rate * Miss Penalty
  - B、IC * CC * IPC
  - C. Told / Tnew
  - D、IC * CC * CPI

- When programming, the memory address used is _____.
  - A. Main memory address.
  - B. Logical address
  - C. Physical address.
  - D. Effective address

- Compared to fully associative cache, the advantage of set associative cache is _____.
  - A. Smaller tag field takes less chip area
  - B. Low rate of block conflict.
  - C. High hit rate.
  - D. High utilization of main memory.

- Assume there are M blocks in a cache, and every K blocks are grouped in one set, then which following description is NOT correct?

A. If K=1, then it's a direct mapped cache.

B. If K=1, then it's a one-way set associative cache.

C. If K=M, then it's a full-associative cache.

D. If K>1 and K<M, then it's a M/K-way set associative cache.

- Assume there is a code segment as following. And the elements in arrays are place in a row-and-row order.

```
for (j = 0; j< 100; j = j+1)
        for (i = 0; i < 5000; i = i+1)
                x[i][j] = x[i][j] + C;      /* C is a constant.  */
```

Some one suggests to optimize the above code by exchanging the nesting of the loops as following:

```
for (i = 0; i < 5000; i = i+1)
    for (j = 0; j< 100; j = j+1)
            x[i][j] = x[i][j] + C;      /* C is a constant.  */
```

- Which of the following statements is correct ?

  A. The optimization can decrease cache misses by improving the spatial locality.

  B. The optimization can decrease cache misses by improving the temporal locality.

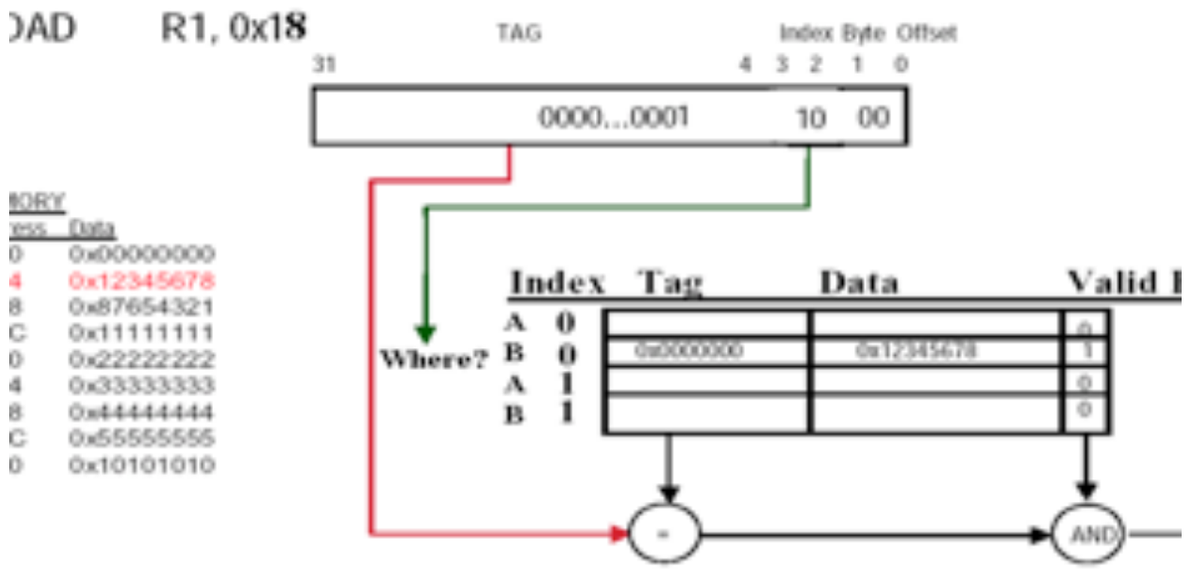
  C. The optimization can decrease cache misses by improving both the temporal locality and spatial locality.

  D. This measurement can not decrease cache misses at all.

- A cache has 64-KB capacity, 128-bytes/line, and is 4-way set-associative. The system containing the cache uses 32-bit addresses.

- The cache has ___ lines.
- The cache has _____ sets.
- Tag information is ___ bits

- In 2-way set-associative cache, assume cache has 4 blocks and each block is 1 word and 2 blocks per set. For instruction LOAD R1, 0x18, is memory access misses? If the access misses, will replacement occur? And where is the location that the new loaded block will be located?



- Memory access to 0x18 will (miss or hit) in the cache .
- Whether there is a replacement (yes or no) .
- Block will be place in Set ___ and Block ___.

- Assume the performance of the basic memory organization is:
  - 4 clock cycles to send the address
  - 56 clock cycles for the access time per word
  - 4 clock cycles to transfer a word of data
- Given a cache block of four words, and that a word is 8 bytes, the miss penalty is _____ (Calculation expression should be given out.) clock cycles, with a memory bandwidth of _____ bytes per clock cycle.

2. (13)Within some **memory/cache** memory hierarchies, there are 2 words in a block. Access time form Cache is 8ns and for main memory miss penalty is 70ns. For the code of C language below, assumes that each element is one word in array (**A[i]**). Except array, another variables has be loaded to registers. While the C codes execute, please calculate and questions below:

```
for ( i=0; i<100; i++)
        s=s+A[i] ;
```

(1) What is the miss rate for data accesses?

(2) What is the average memory access time for data read?

(3) What is the overall CPI including memory access? Assume processor runs at 1.1GHz and has a CPI of 1 excluding memory accesses. Ignores instructions misses and data hazard and control hazard. Assumes assembler code is below:

```
                ……
LOOP:   LOAD    R2,     0(R1)
        ADD     R5,R1,#4
        ADD     R3,     R2,R3          ;s was stored in R3
        BNE     R5,     LOOP
```

- Assume all cache parameters (capacity, associativity, block size) remain fixed except for the single change described in each question. Please provide a brief explanation within three sentences of your answer.

  - (1) “Doubling the block size doubles the number of tage entries in the cache”. Is it true or false ? Why ?

  - (2) “Doubling cache capacity of a direct-mapped cache usually reduces conflict misses.” Is it true or false? Why ?

  - (3) “Doubling cache capacity of a direct-mapped cache usually reduces compulsory misses.” Is it true or false ? why ?

- What's a virtual indexed and physical tagged cache ? What's the advantage of this kind of cache.

- Suppose in a particular cache that uses write-back, that of the data blocks to be swapped out on average 20% are dirty. Suppose the hit time of the cache is 1 cycle, and the miss penalty is 20 cycles for loading the miss block and additional 20 cycles for write back the dirty block. Assume the cache miss rate is 10%.

- (1) What is the speedup up of the memory system if adding a “write-buffer” eliminates 60% of the stall cycles to write back the dirty blocks?

- (2) What is the maximum possible speedup of the overall memory hierarchy of any write-buffer enhancement for this particular cache?

- Consider the following description of a memory hierarchy.
  - Virtual address wide = 45 bits, Memory physical address wide = 38 bits, Page size = 4KB.
  - Cache capacity =8KB. It is a write-through 4-way associative cache. Cache block size = 32B.
- a) How many bits are there in the fields of tag, index and block offset of the physical memory address.
- b) Draw a graph to show a cache line (including tag, data, and some other control bits) in the cache.
- c) Please describe the access procedure to the memory hierarchy system that when a CPU address (virtual address) is given to access data cache

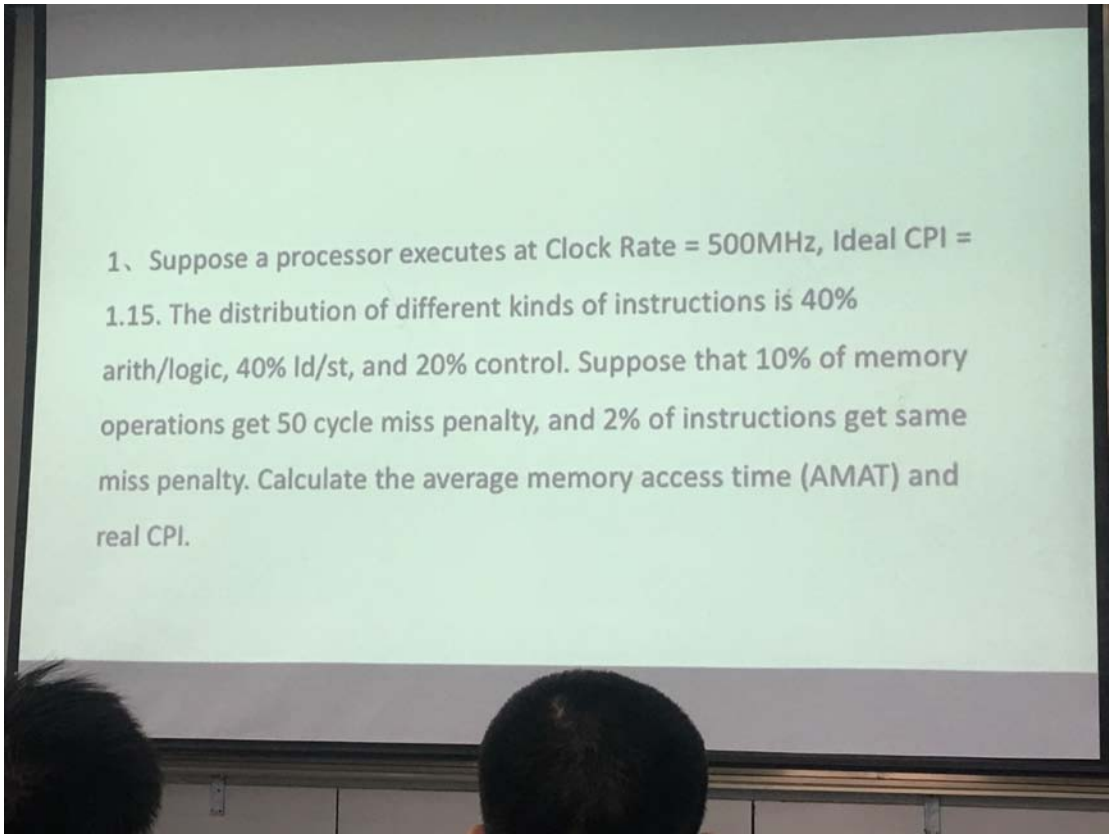
- John is asked to study the effect of set-associativity on the cache performance. Since he now knows the access time of each configuration, he wants to know the miss-rate of each one. For the miss-rate analysis, Ben is considering two small caches: a direct-mapped cache with 8 blocks with 16 bytes/block, and a 4-way set associative cache of the same size. The cache use FIFO replacement and write back strategy with no write allocate scheme.

- John tests the cache by accessing the following sequence of hexadecimal byte addresses, starting with empty caches. For simplicity, assume that the addresses are only 12 bits. Complete the following tables for the direct-mapped cache and both types of 4-way set-associative caches showing the progression of cache contents as accesses occur (in the table, ‘inv’=invalid, and the column of a particular cache line contains the {tag, index} contents of that line). You only need to fill in elements in the table when a value changes, and marked “(D)” if the line is dirty.

Direct-mapped	Line in cache								Hit ? Yes/No
Addres	L0	L1	L2	L3	L4	L5	L6	L7	
R: 0x110	Inv	11	inv	Inv	Inv	inv	Inv	inv	No
R: 0x136				13					No
R: 0x202	20								No
W: 0x133				13(D)					Yes
R: 0x3D2									
W: 0x142									
R: 0x461									
W: 0x204									
R: 0x118									
R: 0x1A4									
R: 0x127									
R: 0x148									
W: 0x142									

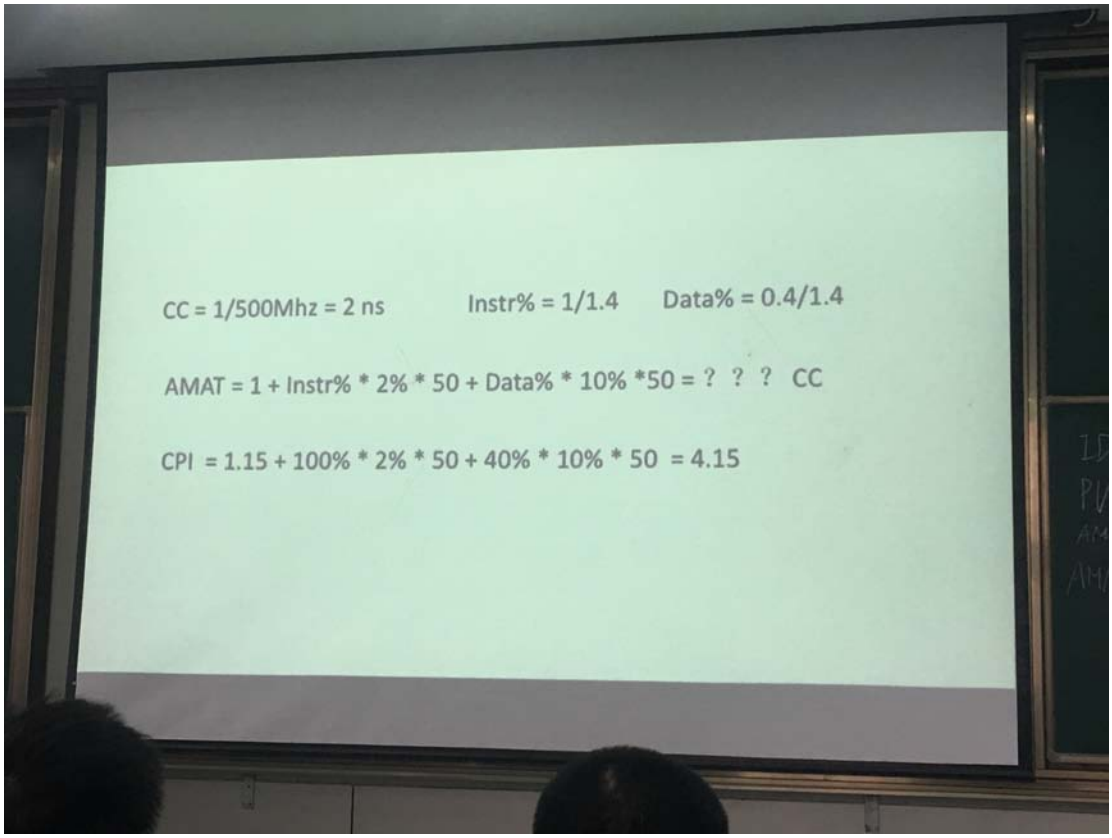
4-way set-Associative	Line in cache								Hit ? Yes/No
	Set0				Set1				
Address	L0	L1	L2	L3	L4	L5	L6	L7	
R: 0x110	Inv	inv	Inv	Inv	11	inv	inv	Inv	No
R: 0x136						13			No
R: 0x202	20								No
W: 0x133						13(D)			Yes
R: 0x3D2									
W: 0x142									
R: 0x461									
W: 0x204									
R: 0x118									
R: 0x1A4									
R: 0x127									
R: 0x148									
W: 0x142									

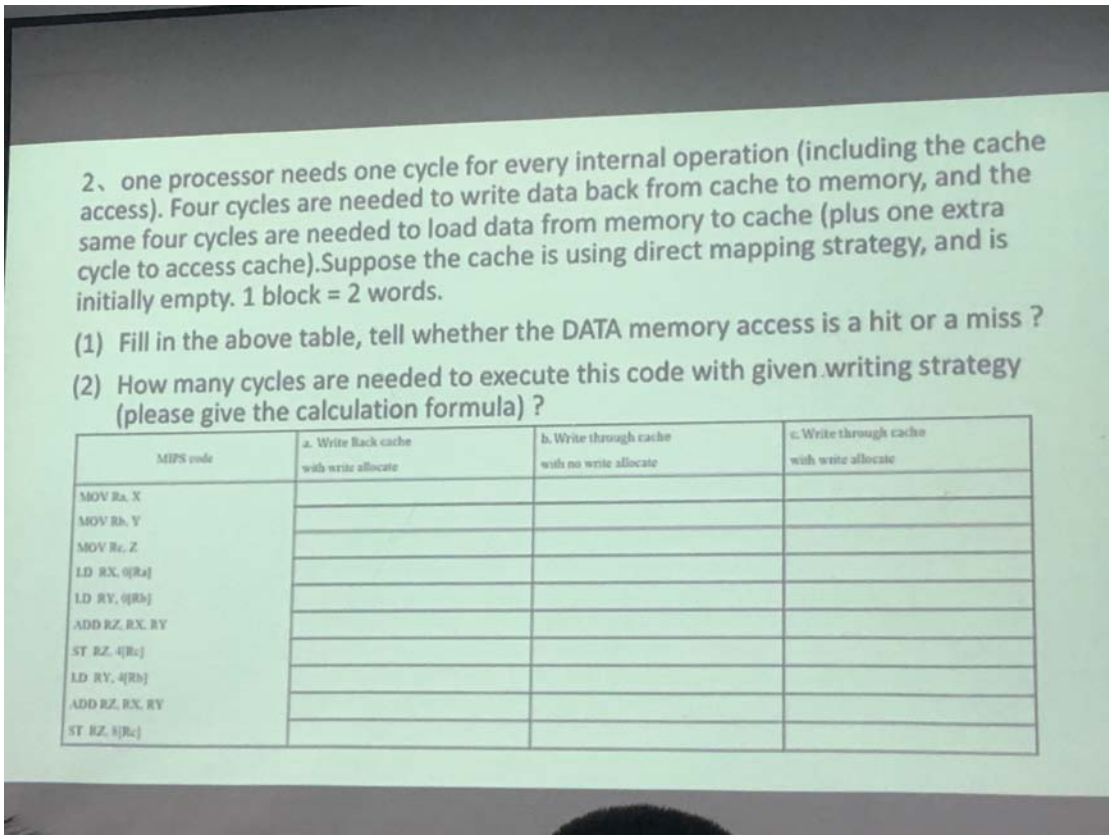
2018 陈文智老师复习



hit time = 1 clock　默认为 1

1 + 100/140*(0.02*50) + 40/140*(0.4*0.1*50)





2、one processor needs one cycle for every internal operation (including the cache access). Four cycles are needed to write data back from cache to memory, and the same four cycles are needed to load data from memory to cache (plus one extra cycle to access cache).Suppose the cache is using direct mapping strategy, and is initially empty. 1 block = 2 words.

(1) Fill in the above table, tell whether the DATA memory access is a hit or a miss ?

(2) How many cycles are needed to execute this code with given writing strategy (please give the calculation formula) ?

MIPS code	a. Write Back cache with write allocate	b. Write through cache with no write allocate	c. Write through cache with write allocate
MOV Ra, X			
MOV Rb, Y			
MOV Rc, Z			
LD RX, 0[Ra]			
LD RY, 0[Rb]			
ADD RZ, RX, RY			
ST RZ, 4[Rc]			
LD RY, 4[Rb]			
ADD RZ, RX, RY			
ST RZ, 8[Rc]			

为什么读的时候是 block 写的时候是 word？读的时候只能是 block ,写的时候可以写 word。

以及 move 的机制 move 因为是寄存器的操作，所以没有涉及到 mem 的操作。

2.CPU 写 Cache 时：

●若 hit，有两种处理方式：

>Write through：把数据同时写到 Cache 和内存中；只写一个 word

>Write back：先把数据写到 Cache 中，再通过 flush 方式写入到内存中。

●若 miss，有两种处理方式：

>Write allocate：先把要写的数据载入到 Cache 中，写 Cache，然后再通过 flush 方式写入到内存中；

>No write allocate：直接把要写的数据写入到内存中。

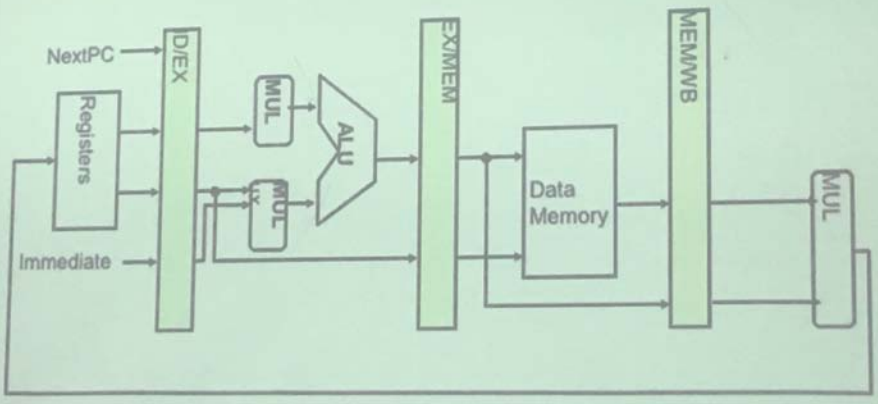
MIPS code	a. Write Back cache with write allocate	b. Write through cache with no write allocate	c. Write through cache with write allocate
MOV Ra, X			
MOV Rb, Y			
MOV Rc, Z			
LD RX, 0[Ra]	Cache Miss	Cache Miss	Cache Miss
LD RY, 0[Rb]	Cache Miss	Cache Miss	Cache Miss
ADD RZ, RX, RY			
ST RZ, 4[Rc]	Cache Miss	Cache Miss	Cache Miss
LD RY, 4[Rb]	Cache hit	Cache hit	Cache hit
ADD RZ, RX, RY			
ST RZ, 0[Rc]	Cache Miss	Cache Miss	Cache Miss

(a) 10CC + 4 Miss * (4cc * 2 word+1cc) = 46 CC

(b) 10CC + 2 load miss * (4cc * 2 word+1cc) + 2 write miss * (4CC * 1 word) = 36 CC

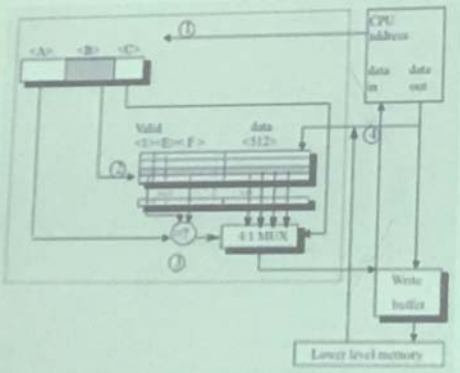
(c) 10CC + 2 laod miss * (4cc * 2 word+1cc) + 2 write miss * [(4cc * 2 word +1) + (4CC * 1 word)] = 52 CC

3、forwarding



4、Considering the following description of a one-way set-associative write-back cache .

Memory physical address wide = 32 bits, Cache capacity =4KB, Block size=64Byte



A is called _____ in ____ bits.
B is called _____ in ____ bits.
C is called _____ in _____ bits.

E is called _____ only 1 bit.
F is called _____in _____bits.

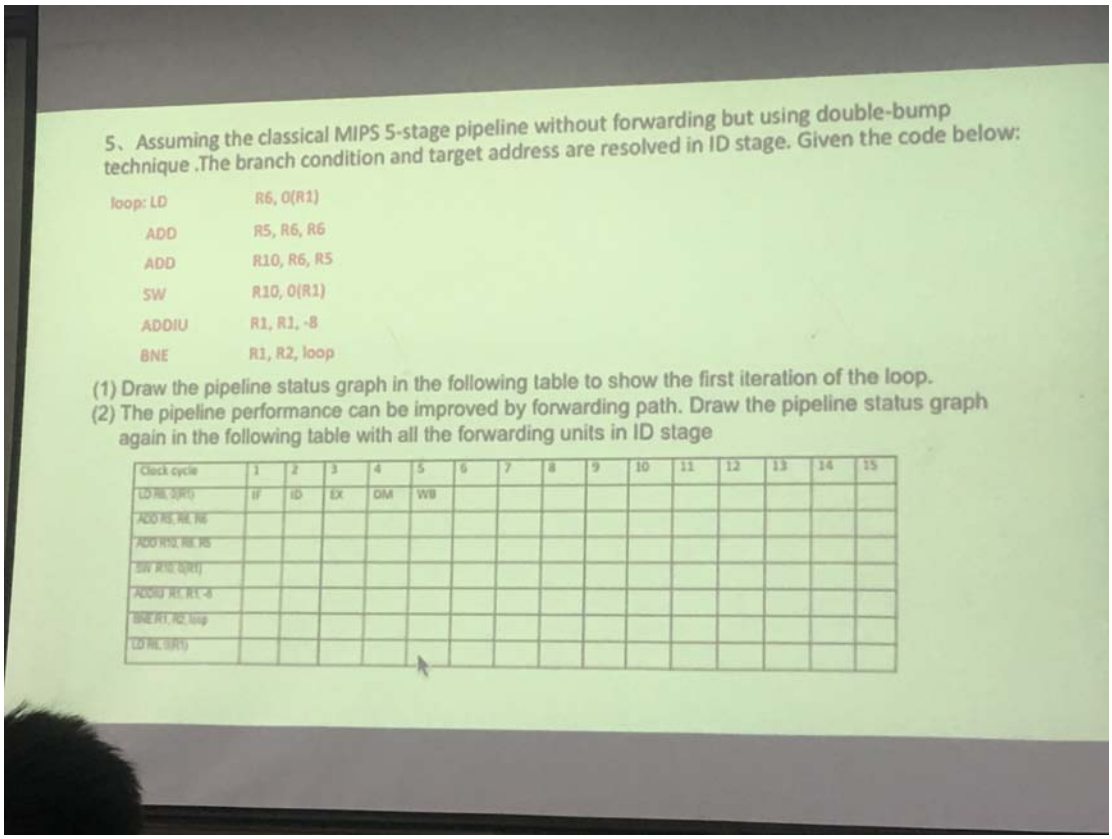
A tag 32-6-6

B index 4KB / 64B = 2^6 = 6bits

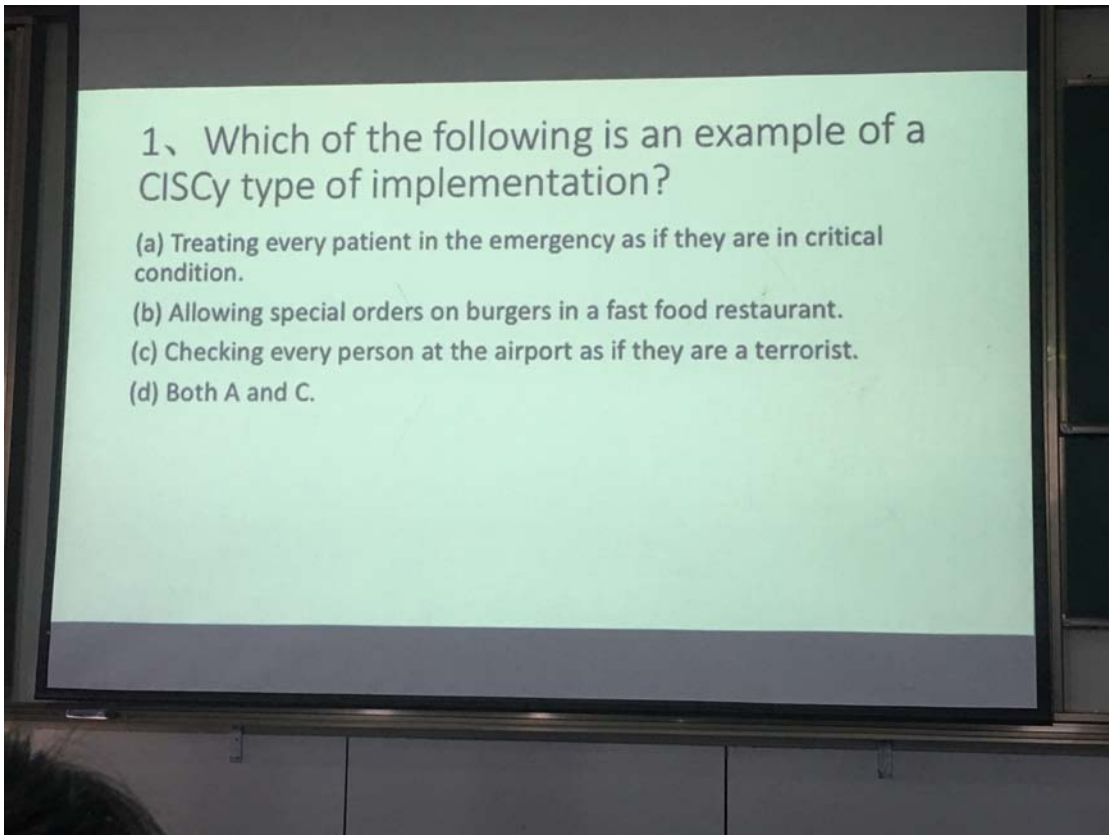
C offset 6bits

E dirty 1

F tag 20bits



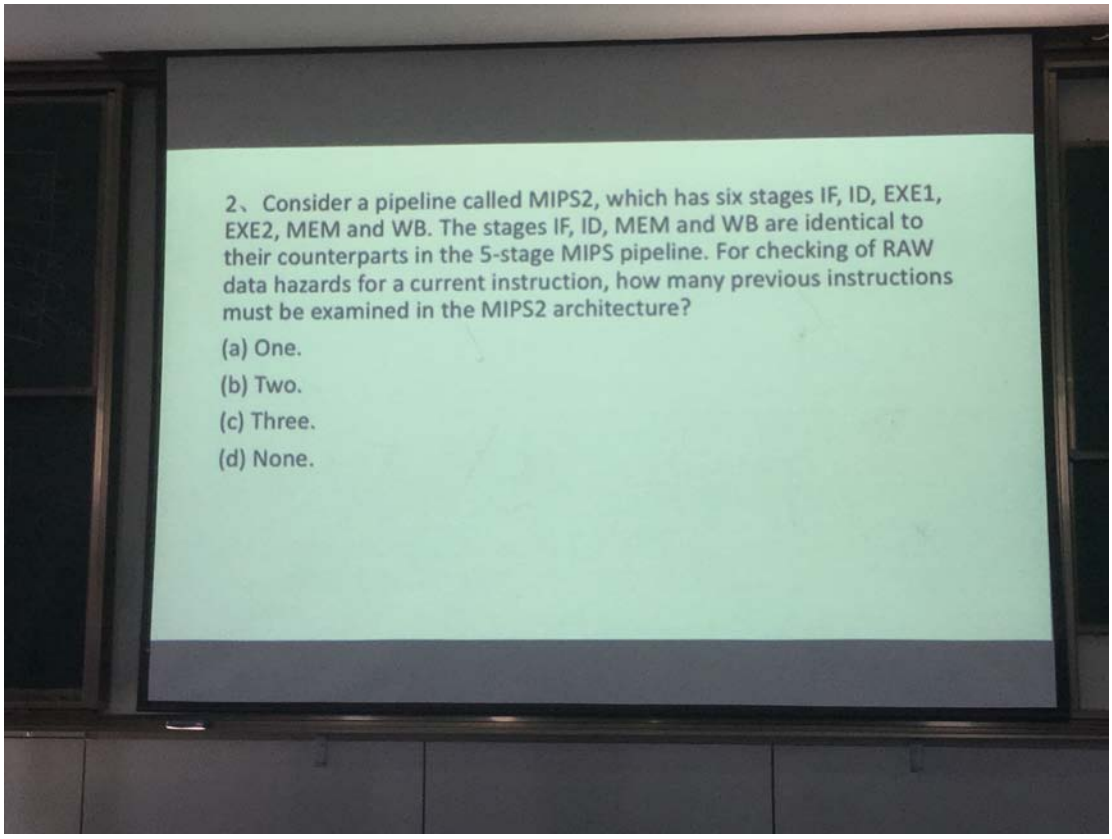
前面有 S 后面紧跟 S，ID 的位置可以在中间的某个位置



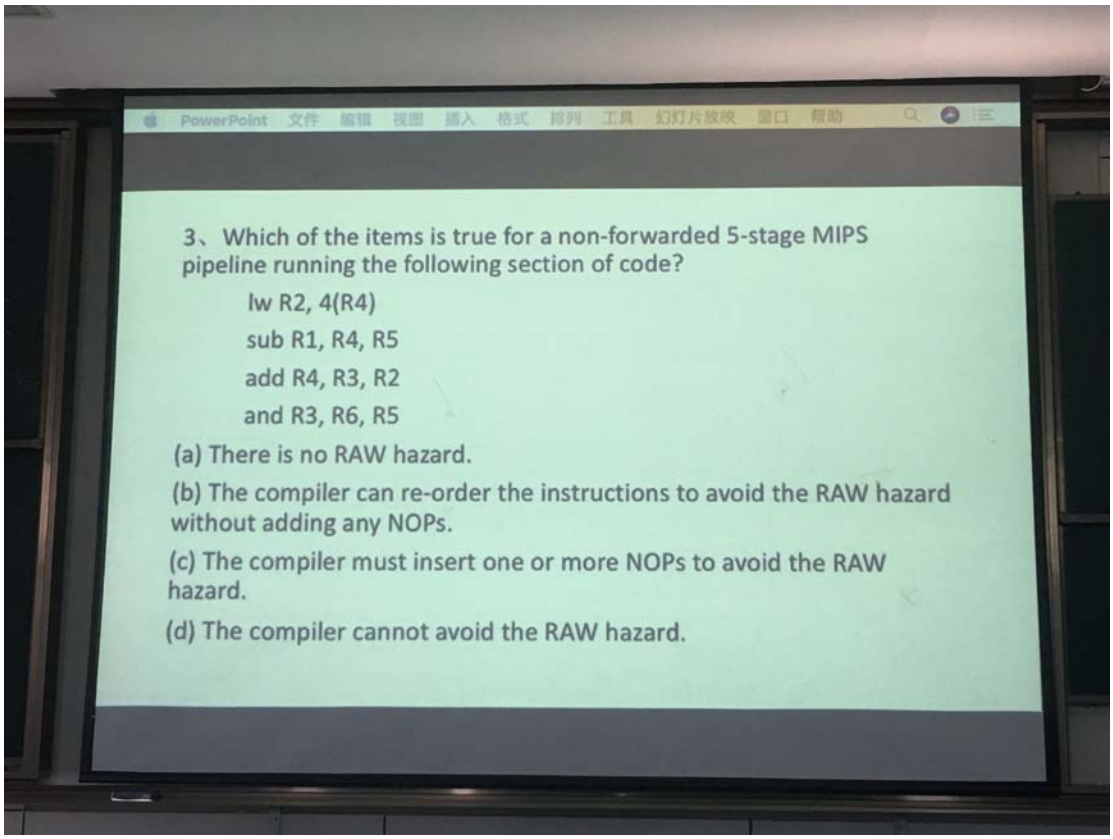
选择 B，计算机中原来的指令从精简指令集到复杂指令集，然后又回到精简。

Complex Instruction Set Computing 缩写：CISC

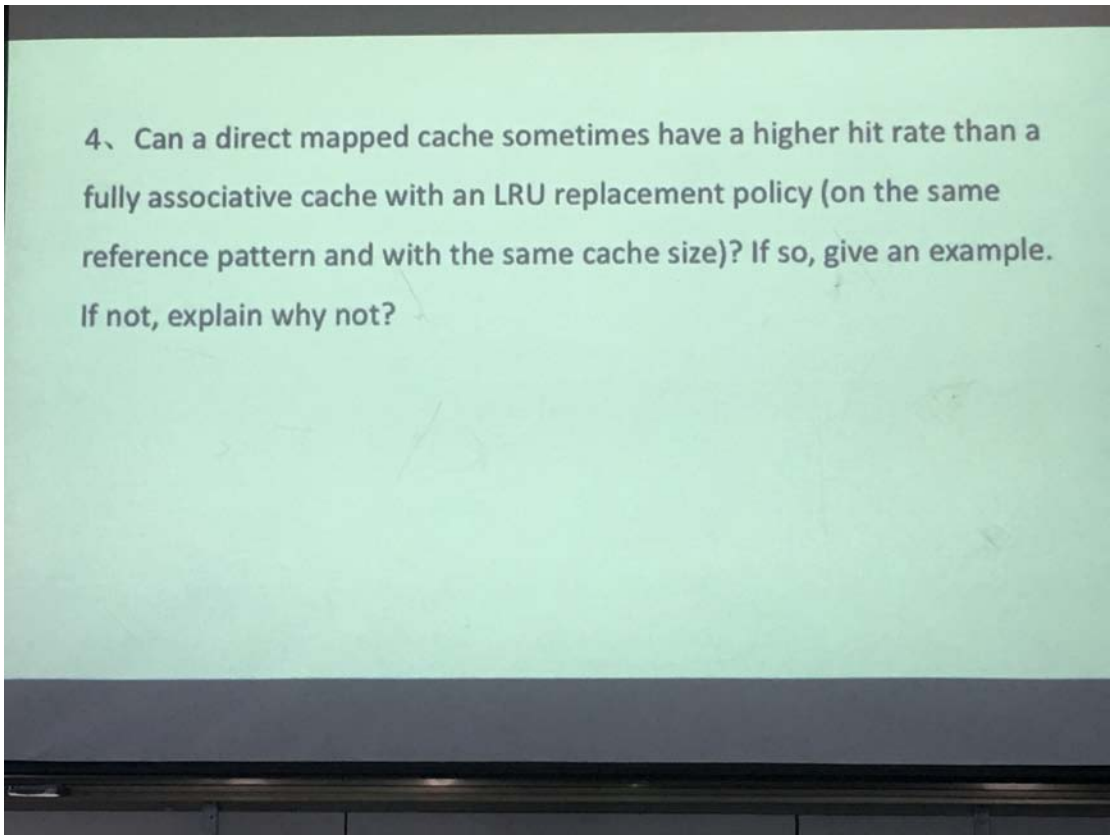
reduced instruction set computing，缩写：RISC



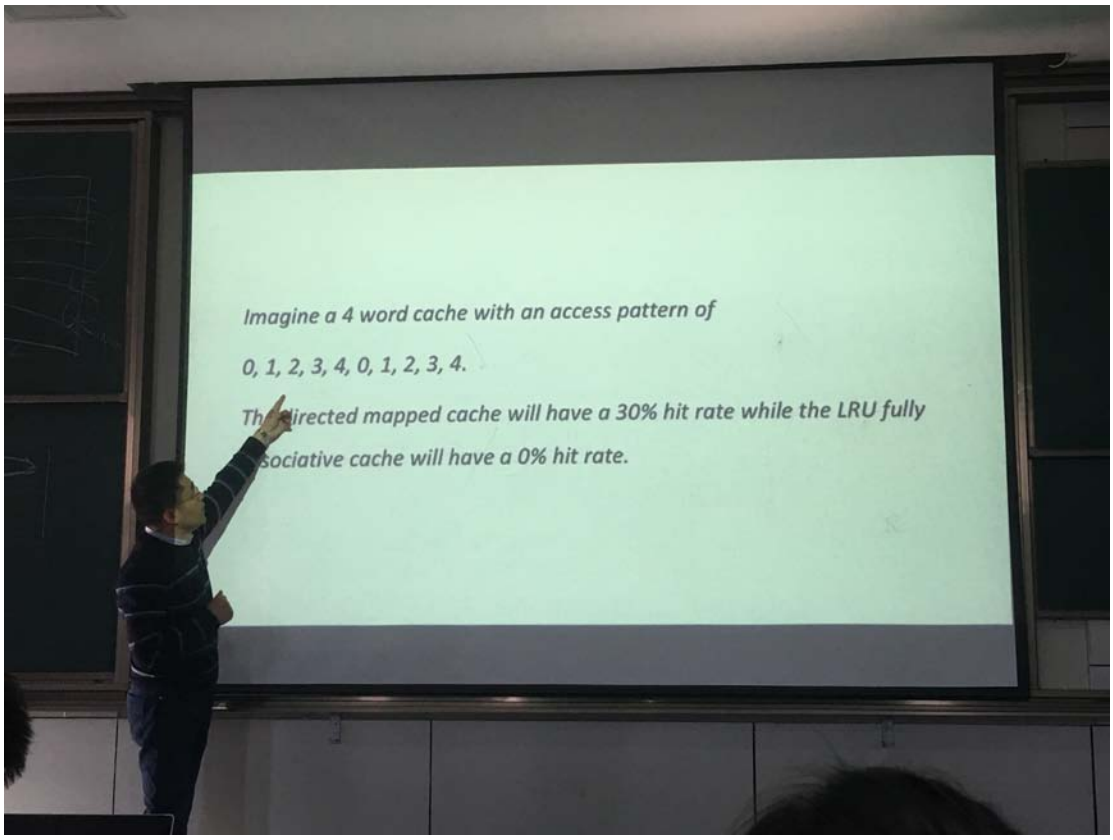
选择 C，？？？？？

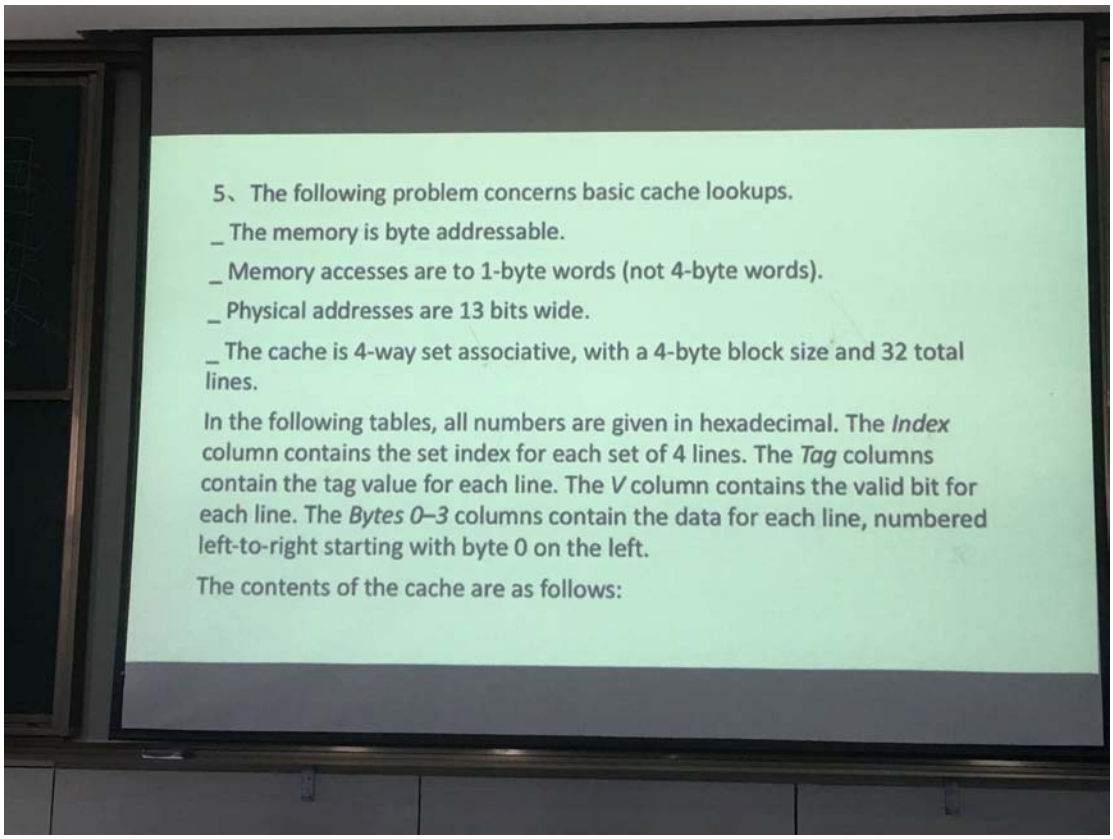


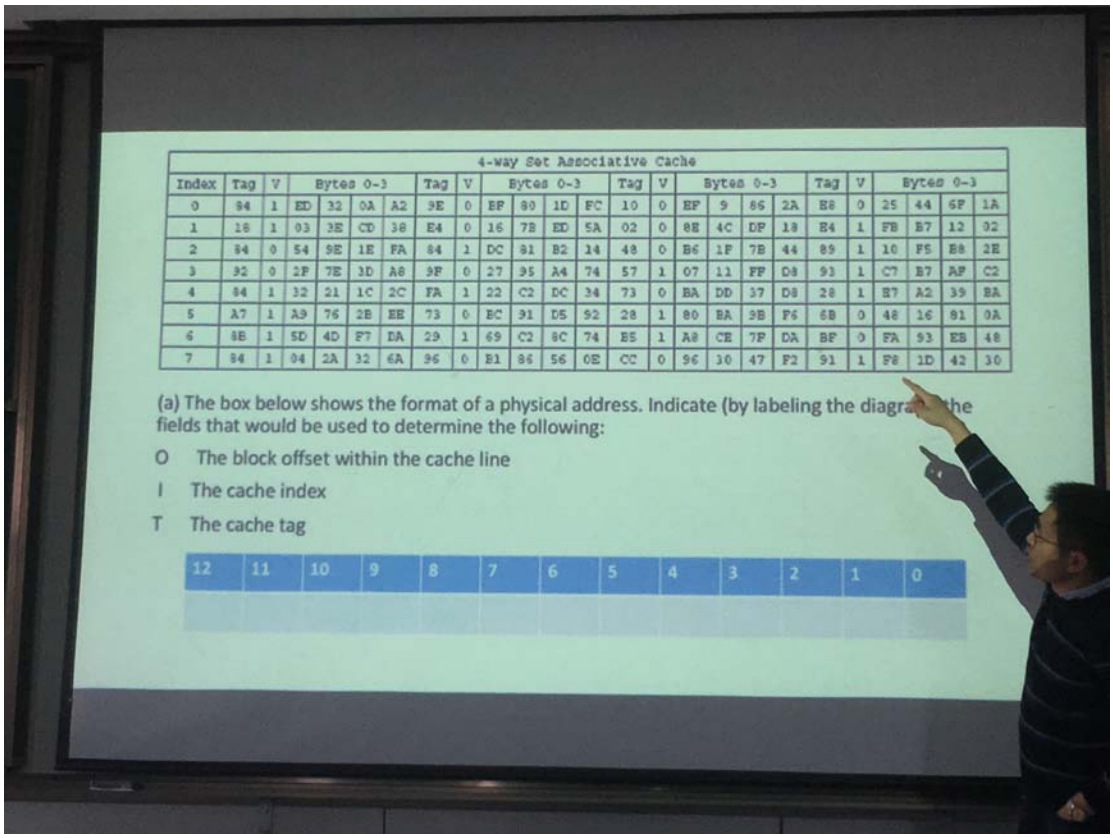
选 C，第一条和第三条冲突。


4、Can a direct mapped cache sometimes have a higher hit rate than a fully associative cache with an LRU replacement policy (on the same reference pattern and with the same cache size)? If so, give an example. If not, explain why not?

可以使用 1、2、3、4、5


Imagine a 4 word cache with an access pattern of
0, 1, 2, 3, 4, 0, 1, 2, 3, 4.
Th irected mapped cache will have a 30% hit rate while the LRU fully
sociative cache will have a 0% hit rate.

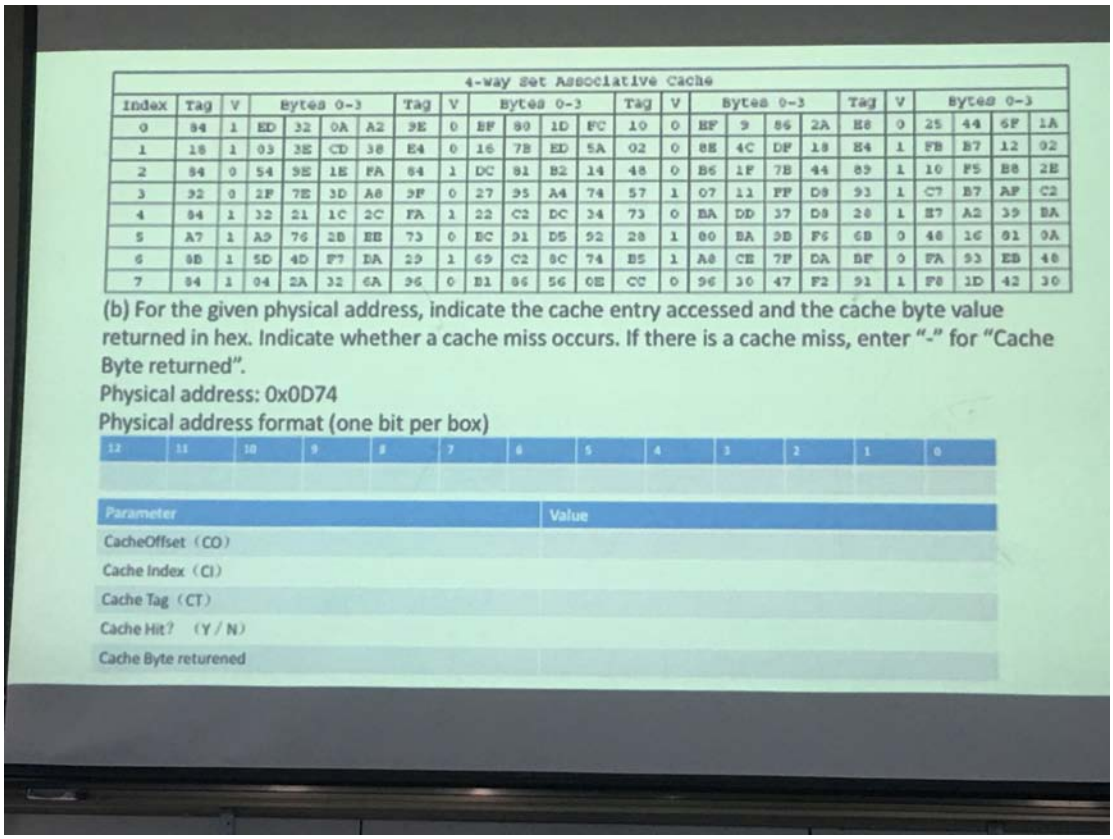




Index	Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3			
0	84	1	ED	32	0A	A2	3E	0	BF	80	1D	FC	10	0	EF	9	86	2A	E8	0	25	44	6F	1A
1	16	1	03	3E	CD	38	E4	0	16	7B	ED	5A	02	0	0E	4C	DF	18	E4	1	FB	B7	12	02
2	84	0	54	9E	1E	FA	84	1	DC	81	B2	14	48	0	B6	1F	7B	44	89	1	10	F5	B8	2E
3	92	0	2F	7E	3D	A8	9F	0	27	95	A4	74	57	1	07	11	FF	D8	93	1	C7	B7	AF	C2
4	84	1	32	21	1C	2C	FA	1	22	C2	DC	34	73	0	BA	DD	37	D8	28	1	E7	A2	39	BA
5	A7	1	A9	76	2B	EE	73	0	EC	91	D5	52	28	1	80	BA	9B	F6	6B	0	48	16	81	0A
6	8B	1	5D	4D	F7	DA	29	1	69	C2	8C	74	B5	1	A8	CE	7F	DA	BF	0	FA	93	EB	48
7	84	1	04	2A	32	6A	96	0	B1	86	56	0E	CC	0	96	30	47	F2	91	1	F8	1D	42	30

0-1offset 2-4index 5-12tag

offset 索引到 byte



4-way Set Associative Cache

Index	Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3			
0	84	1	ED	32	0A	A2	9E	0	BF	80	1D	FC	10	0	BF	9	86	2A	E8	0	25	44	6F	1A
1	18	1	03	3E	CD	38	E4	0	16	7B	ED	5A	02	0	8E	4C	DF	18	E4	1	FB	B7	12	02
2	84	0	54	9E	1E	FA	84	1	DC	81	B2	14	48	0	B6	1F	7B	44	89	1	10	F5	B8	2E
3	92	0	2F	7E	3D	A8	9F	0	27	95	A4	74	57	1	07	11	FF	D9	93	1	C7	B7	AF	C2
4	84	1	32	21	1C	2C	FA	1	22	C2	DC	34	73	0	BA	DD	37	D8	28	1	E7	A2	39	BA
5	A7	1	A9	76	2B	EE	73	0	BC	91	D5	92	28	1	80	BA	3B	F6	6B	0	48	1C	81	9A
6	9B	1	5D	4D	F7	DA	29	1	69	C2	8C	74	B5	1	A8	CE	7F	DA	BF	0	FA	93	ED	48
7	84	1	04	2A	32	6A	96	0	B1	86	56	0E	CC	0	96	30	47	F2	91	1	F8	1D	42	30

(b) For the given physical address, indicate the cache entry accessed and the cache byte value returned in hex. Indicate whether a cache miss occurs. If there is a cache miss, enter "-" for "Cache Byte returned".

Physical address: 0x0D74

Physical address format (one bit per box)

12	11	10	9	8	7	6	5	4	3	2	1	0

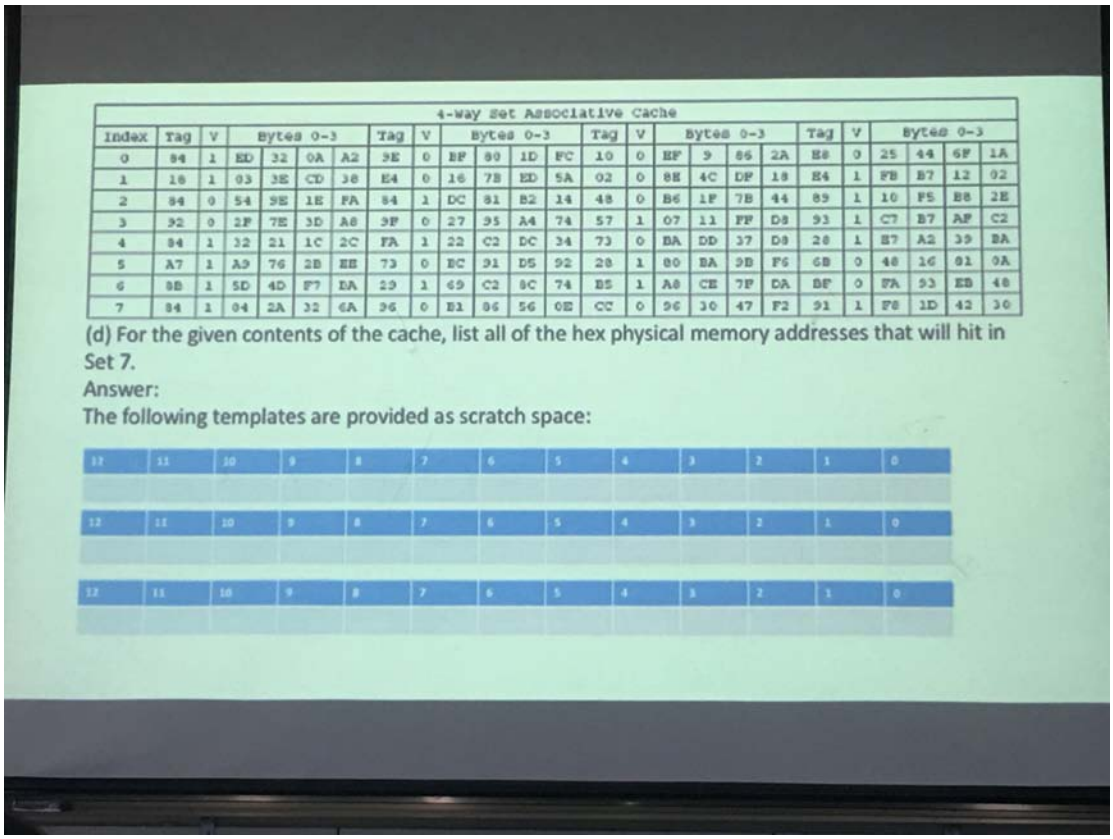
Parameter	Value
CacheOffset (CO)	
Cache Index (CI)	
Cache Tag (CT)	
Cache Hit? (Y / N)	
Cache Byte returened	

110 1011 101 00

offset: 0 index: 5 tag: 6B valid: 0

所以失配了

如果是 0AEE 则找到 FF



4-way Set Associative Cache

Index	Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3			
0	84	1	ED	32	0A	A2	9E	0	BF	80	1D	FC	10	0	EF	9	86	2A	E8	0	25	44	6F	1A
1	16	1	03	3E	CD	38	E4	0	16	7B	ED	5A	02	0	8E	4C	DF	18	E4	1	FB	B7	12	02
2	84	0	54	9E	1E	FA	84	1	DC	81	B2	14	48	0	B6	1F	7B	44	89	1	10	F5	B8	2E
3	92	0	2F	7E	3D	A8	9F	0	27	95	A4	74	57	1	07	11	FF	D8	93	1	C7	B7	AF	C2
4	84	1	32	21	1C	2C	FA	1	22	C2	DC	34	73	0	BA	DD	37	D8	28	1	E7	A2	39	BA
5	A7	1	A9	76	2B	EE	73	0	BC	91	D5	92	28	1	80	BA	9B	F6	6B	0	48	16	81	9A
6	8B	1	5D	4D	F7	DA	29	1	69	C2	9C	74	B5	1	A8	CE	7F	DA	BF	0	FA	53	EB	48
7	84	1	04	2A	32	6A	96	0	B1	06	56	0E	CC	0	96	30	47	F2	91	1	F8	1D	42	30

(d) For the given contents of the cache, list all of the hex physical memory addresses that will hit in Set 7.

Answer:

The following templates are provided as scratch space:

12	11	10	9	8	7	6	5	4	3	2	1	0

12	11	10	9	8	7	6	5	4	3	2	1	0

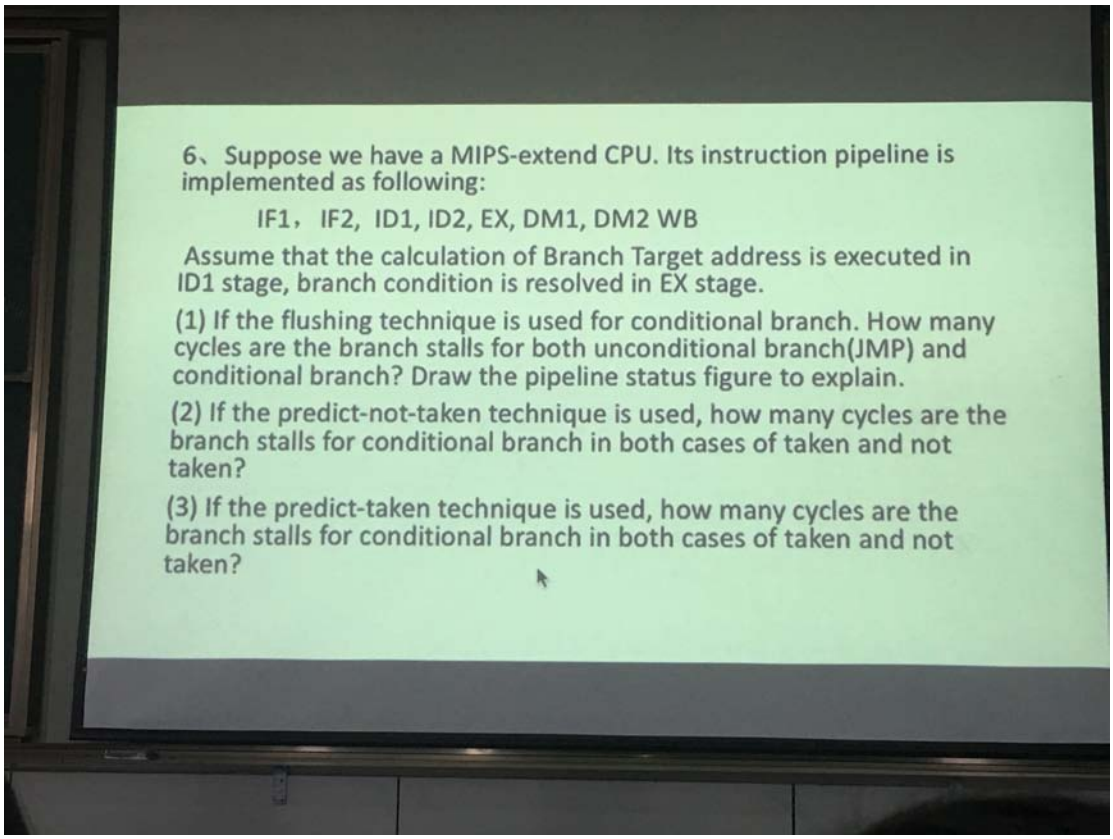
12	11	10	9	8	7	6	5	4	3	2	1	0

1000 0100 111 00

1001 0001 111 00

109C 109D 109E 109F

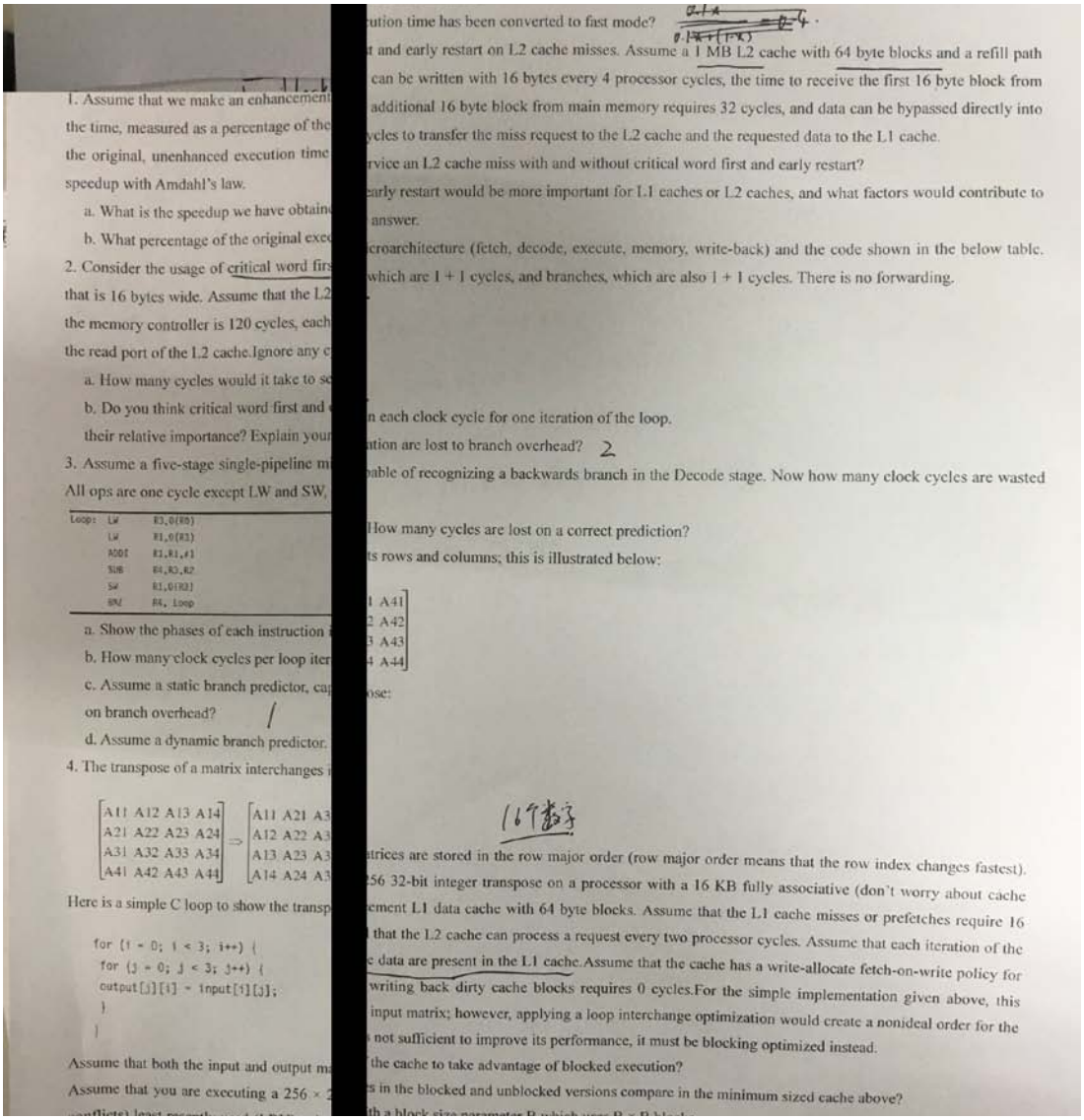
123C 123D 123E 123F



(1) Jump 2 branch 4

(2) Not Taken 0 taken 4

(3) Taken 2 not taken 4



1. Assume that we make an enhancement
the time, measured as a percentage of the
the original, unenhanced execution time
speedup with Amdahl's law.
a. What is the speedup we have obtain
b. What percentage of the original exe
2. Consider the usage of critical word firs
that is 16 bytes wide. Assume that the L2
the memory controller is 120 cycles, each
the read port of the L2 cache.Ignore any c
a. How many cycles would it take to s
b. Do you think critical word first and
their relative importance? Explain you
3. Assume a five-stage single-pipeline m
All ops are one cycle except LW and SW,

```
Loop:  LW     R3,0(R5)
       LW     R1,0(R1)
       ADDI   R1,R1,#1
       SUB    R4,R3,R2
       SW     R1,0(R2)
       BNZ    R4, Loop
```

a. Show the phases of each instruction
b. How many clock cycles per loop iter
c. Assume a static branch predictor, ca
on branch overhead?
d. Assume a dynamic branch predictor,
4. The transpose of a matrix interchanges

$$\begin{bmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \\ A41 & A42 & A43 & A44 \end{bmatrix} \Rightarrow \begin{bmatrix} A11 & A21 & A3 \\ A12 & A22 & A3 \\ A13 & A23 & A3 \\ A14 & A24 & A3 \end{bmatrix}$$

Here is a simple C loop to show the transp

```
for (i = 0; i < 3; i++) {
 for (j = 0; j < 3; j++) {
 output[j][i] = input[i][j];
 }
}
```

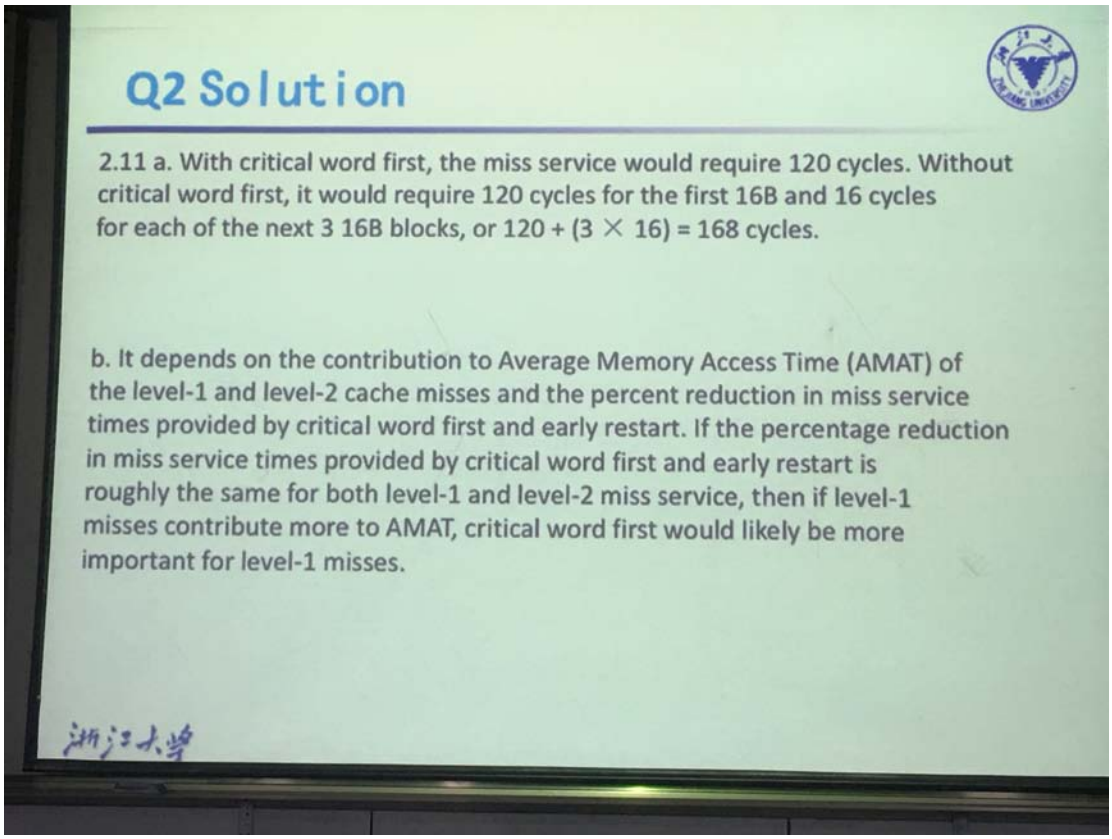
Assume that both the input and output m
Assume that you are executing a 256 ×

ution time has been converted to fast mode?
t and early restart on L2 cache misses. Assume a 1 MB L2 cache with 64 byte blocks and a refill path
can be written with 16 bytes every 4 processor cycles, the time to receive the first 16 byte block from
additional 16 byte block from main memory requires 32 cycles, and data can be bypassed directly into
ycles to transfer the miss request to the L2 cache and the requested data to the L1 cache.
rvice an L2 cache miss with and without critical word first and early restart?
arly restart would be more important for L1 caches or L2 caches, and what factors would contribute to
answer.
croarchitecture (fetch, decode, execute, memory, write-back) and the code shown in the below table.
which are 1 + 1 cycles, and branches, which are also 1 + 1 cycles. There is no forwarding.
n each clock cycle for one iteration of the loop.
ution are lost to branch overhead? 2
able of recognizing a backwards branch in the Decode stage. Now how many clock cycles are wasted
How many cycles are lost on a correct prediction?
ts rows and columns; this is illustrated below:
1 A41
2 A42
3 A43
4 A44
ose:
16个数字
trices are stored in the row major order (row major order means that the row index changes fastest).
256 32-bit integer transpose on a processor with a 16 KB fully associative (don't worry about cache
ement L1 data cache with 64 byte blocks. Assume that the L1 cache misses or prefetches require 16
that the L2 cache can process a request every two processor cycles. Assume that each iteration of the
e data are present in the L1 cache.Assume that the cache has a write-allocate fetch-on-write policy for
writing back dirty cache blocks requires 0 cycles.For the simple implementation given above, this
input matrix; however, applying a loop interchange optimization would create a nonideal order for the
not sufficient to improve its performance, it must be blocking optimized instead.
the cache to take advantage of blocked execution?
s in the blocked and unblocked versions compare in the minimum sized cache above?

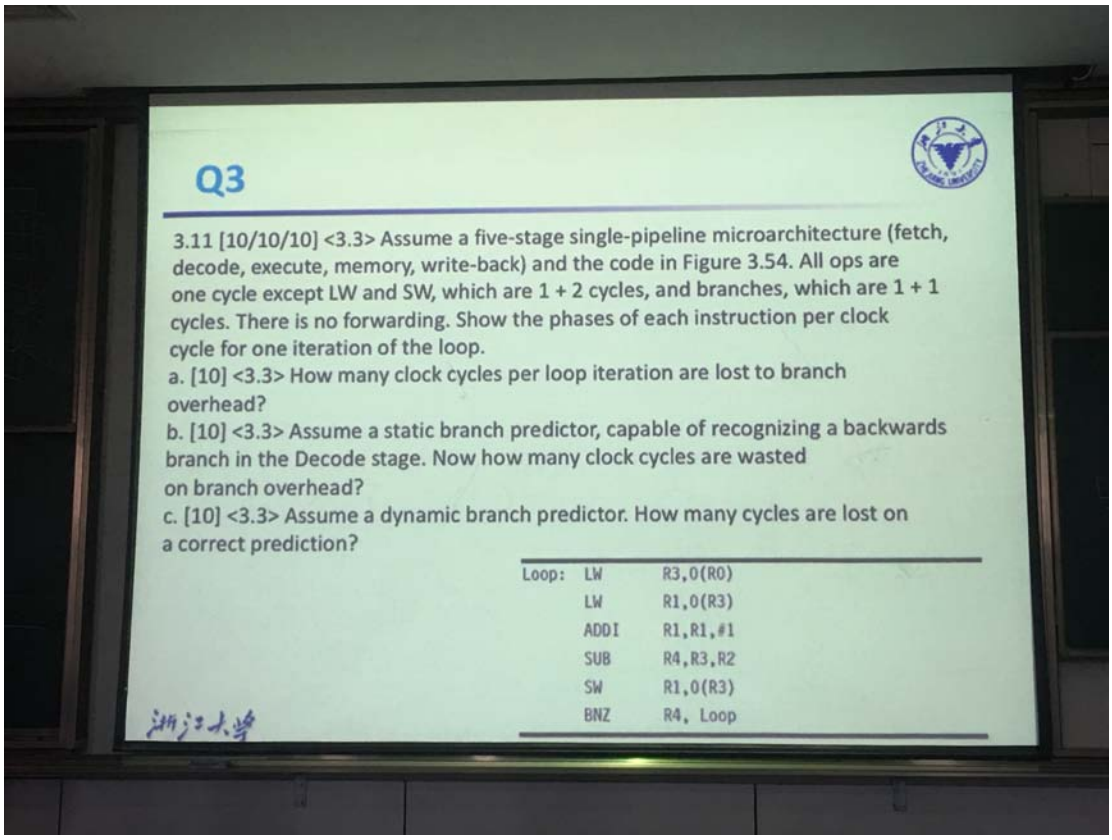
第二题

如果直接取 120 因为有旁路可以过去

120+16*3 = 168



在 L1 和 L2 的失配损失中作用有多大？如果贡献度一样的话……



- To solve the data hazard in the following instructions, we must_____.

  LD R2, 0(R3): IF ID ① EX ② MEM ③ WB

  ADD R1, R5, R2: IF ④ ID ⑤ EX ⑥ MEM WB

- A. Bypassing from ③ to ⑤;
- B. Bypassing from ② to ⑤;
- C. Insert a stall in ⑤;
- D. Bypassing from ② to ⑥;

- To solve the data hazard in the following instructions, the bypassing from _____ to _____ is needed.

ADD R2, R3, R5: IF ID ① EX② MEM ③ WB

SUB R1, R4, R2: IF ④ ID ⑤ EX⑥ MEM WB

- A. ② , ⑤ B. ① , ⑥ C. ③ , ⑥ D. ① , ⑤
  E. ②, ⑥

- For the following code sequence, which kind of forwarding is not taken?

```
DADD R1,R2,R3

LD R4,0(R1)

SD R4,12(R1)
```

- A. EX/MEM.ALUoutput ALUinput
- B. MEM/WB.ALUoutput ALUinput
- C. MEM/WB.LMD DMinput
- D. MEM/WB.LMD ALUinput

- To solve the control hazard in following instructions, ____ is the best choice to be put into the delay slot.

ADD R3, R1, R2----------①

BNEZ R1, DES

< Delay Slot >

SUB R5, R4, R6----------②

DES: SUB R7, R9, R8----------③

- A. It depends B. ③ C. ② D. ①. E. None

- If the following two instructions are executed on the five-stage MIPS pipeline and the branch instruction is resolved in ID stage, then which of the following statement is correct ?

```
SUB    R1, R2, R3          ;R1 R2 – R3

BNEZ  R2, L1           ;if R2 <> 0, go to L1
```

- A. There is only a control stall.
- B. There are a data hazard stall and a control stall
- C. There is only a data hazard stall
- D. No stall at all.

21. Which sequence of instructions cannot use forwarding path to address all potential data hazards?

A. Lw R1, 0(R2) And R6, R1, R7 Or R8, R1, R9	B. Lw R1, 0(R2) Sub R6, R2, R7 Or R8, R6, R9	C. Add R1, R2, R3 Lw R4, 0(R1)	D. Add R1, R2, R3 SD R4, 12(R1)

3. (13) Consider the following pipeline. All instructions have five cycles but autoincrement addressing instruction, which is IF, ID, EX, MEM, WB. Branch will complete at the third cycle. The pipeline extended MIPS pipeline in autoincrement addressing mode which have seven pipe stages. The Fig 1 is an example in autoincrement addressing:

The register files can perform two reads and two write every clock cycle. To handle reads and writes to the same register, assume the register write in the first half of the clock cycle and the read in the second half.

Read the following code segment . The pipeline has forwarding path.

```
Loop:   LW R1, 4(R2)
        ADD R2, (R1)+
        ADDI R3, R3, #4
        SUB R4, R1, R2
        SW R2, 4(R4)+
        BNEZ R3, Loop
```

(question1)if the pipeline has no delay slot, find out how forwarding path work in every instruction? Draw the pipe stage diagram, mark the every forwarding path in diagram.

eg: Add R1, (R2)+

means: Regs[R2]←Regs[R2]+4

Regs[R1]←Regs[R1] + Mem[Regs[R2]]

Fig 1: example instruction of autoincrement addressing

IF: Instruction fetch

ID: Instruction decode

ADDR: AutoIncrement Addressing

WB1: Write Result of ADDR to Register file

EX: Memory Reference: Calculate the absolute address

ALU Instruction: Calculate.

MEM: Memory Access

WB2: Write Result to Register file

Fig 2: seven pipeline stage of the autoincrement addressing

(question2)if the pipeline has one delay slot, how to adjust the code segment. Draw the pipe stage diagram.

四、**(10 points)** Consider the following instruction sequence. An equivalent sequence of C-like pseudocode is also provided.

```
I1:  L.D     F1,    0(R1)        ; F1= *r1;
I2:  Mul.D   F2, F0, F2           ; F2 = F0* F2;
I3:  ADD.D   F1, F2, F2           ; F1= F2 + F2
I4:  LD      F2,    0(R2)        ; F2 = *R2
I5:  ADD.D   F3, F1, F2           ; F3 = F1+F2
I6:  S.D     F3, 0(R3)            ; *R3 = F3
```

Fill out the table below to identify all RAW, WAW, WAR dependences in the above sequence. Do not worry about memory dependencies for this question. The dependencies for thie question. The dependency between I2 and I3 is already filled for you.

Current instruction \ Ealier instruction (older)	I1	I2	I3	I4	I5	I6
I1	_					
I2		_				
I3		RAW	_			
I4				_		
I5					_	
I6						_

五、**(15points)** Assuming the classical MIPS 5-stage pipeline without forwarding except that in the second half of the WB, the value written to a register may be read. The branch condition and target address are resolved in ID stage. Given the code below:

```
loop: LD R6, 0(R1)
      ADD R5, R6, R6
      ADD R10, R6, R5
      ADD R8, R8, R6
      ADDIU R1, R1, -8
      BNE R1, R2, loop
```

(1) (5points) Draw the pipeline status graph in the following table to show the first iteration of the loop.

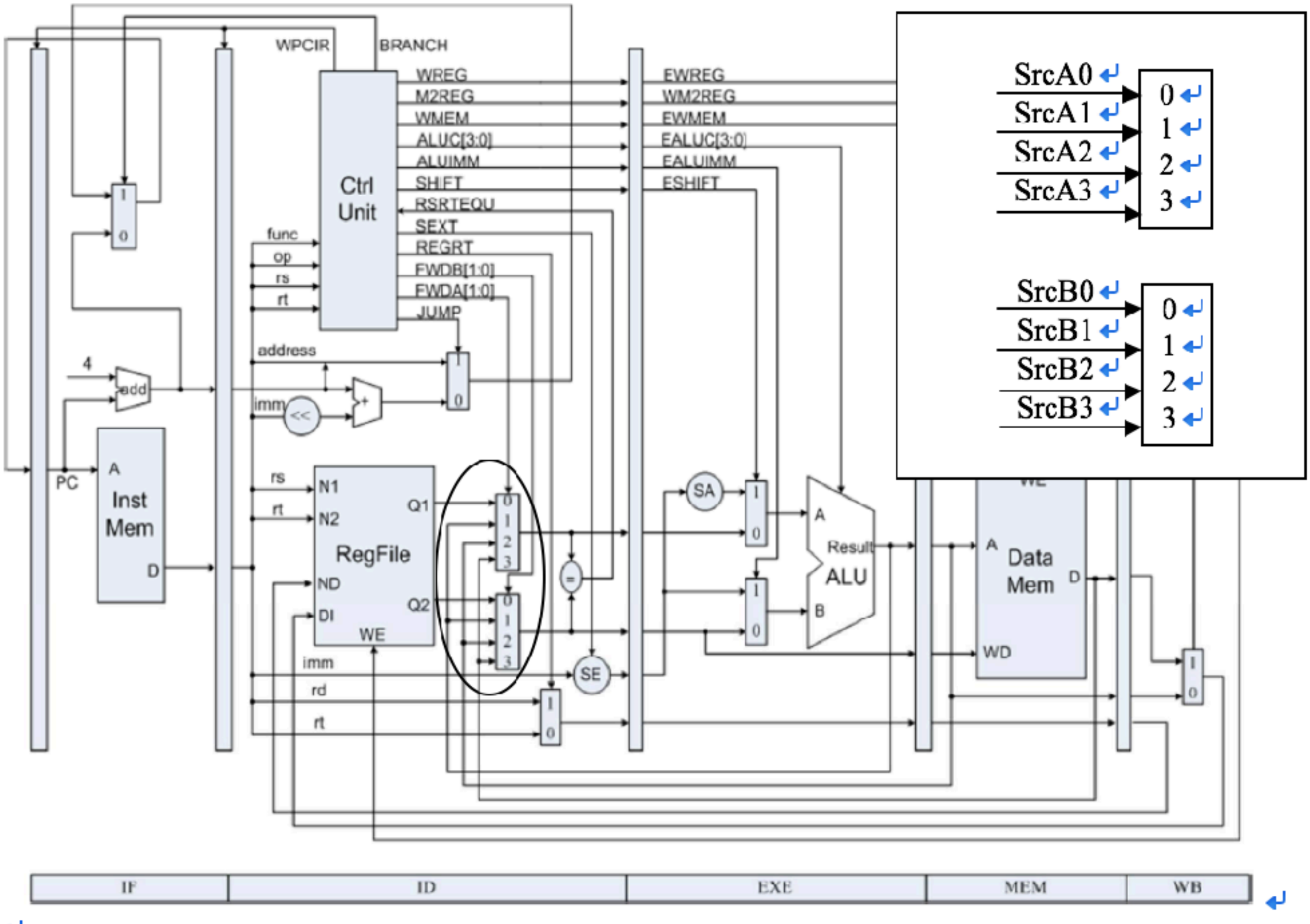
Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LD R6, 0(R1)	IF	ID	EX	DM	WB										
ADD R5, R6, R6															
ADD R10, R6, R5															
ADD R8, R8, R6															
ADDIU R1, R1, -8															
BNE R1, R2, loop															
LD R6, 0(R1)															

(2) (5 points) The pipeline performance can be improved by forwarding path. Draw the pipeline status graph again in the following table with all the forwarding units in ID stage (as that in question 七).

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LD R6, 0(R1)	IF	ID	EX	DM	WB										
ADD R5, R6, R6															
ADD R10, R6, R5															
ADD R8, R8, R6															
ADDIU R1, R1, -8															
BNE R1, R2, loop															
LD R6, 0(R1)															

(3) (5 points) Explain how delayed branch technique improves the pipeline performance in general. Write the updated code after compiler scheduling. Underline the instruction in the delayed slot.

六、**(10 points)**Assume that a MIPS 5-stage pipeline is implemented as that in the figure below. We enlarge the data path in the ellipse as below.



(1) Fill in the blanks in the following table. Indicate whether each of the following instruction sequences cause a stall in the pipeline and which datapath (SrcA0-3, SrcB0-3) is used for forwarding the result. (2) Fill in the last line, give an example instruction sequence that results no stall and using the indicated data path. Consider each sequence seperately and assume that the pipeline is initially idle(for example, it has been executing nothing but nop instructions). Registers involved in inter-instruction dependencies are highlighted in bold for your convenience.

Instruction sequence	Stalls (Yes or No)	Which datapath
Add ***R1***, R2, R3 Lw R4, 0(***R1***)	No	SrcA1

Add ***R1***, 0(R2)
Add R3, R4, R5
Lw R5, 0(***R1***)		

Lw ***R1***, 0(R2) Lw R3, 0(***R1***)		
Lw ***R1***, 0(R2) Sw ***R1***, 0(R3)		
Add ***R3***, R1, R4, Sw ***R3***, 0(R2)		,
	No	SrcB2

三、（10 points）. Suppose we are considering to improve the system performance by optimizing the compiler for a load-store computer of 1GHz clock rate. Measurements of the load-store machine showing the instruction mix and clock cycle counts per instruction before optimization are given in the following table. Assume that the optimized compiler can reduce 50% of the ALU (Arithmetic Logic Unit) operations while other instructions are kept unchanged.

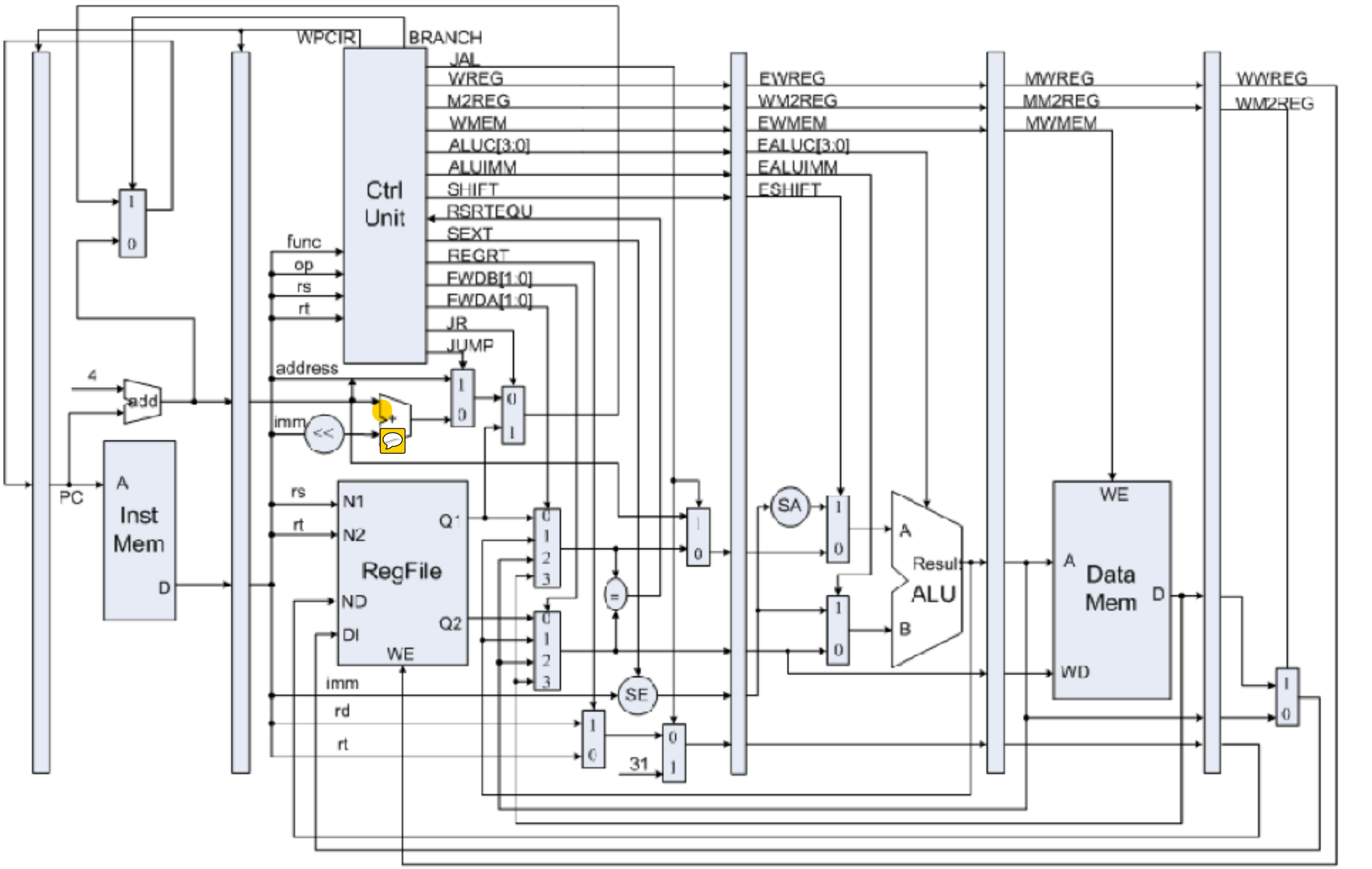
Operation	Instruction	CPI
ALU	430000	1
Loads	210000	2
Stores	120000	2
Branch	240000	2

(1). Calculate average CPI before and after optimization.

(2). What's the overall speedup gained by compiler optimization?

(3). If memory-stalls are 200000 clock cycles for the whole program, what is the impact on performance when the memory system is included? (Calculate the CPU time considering the memory-stalls.)

六、(22 points) One MIPS five-stage-pipelined CPU, which can support 15 MIPS instructions including ADD、SUB、AND、OR、SLT、ADDI、ANDI、ORI、LW、SW、BNE、BEQ、JUMP、JAL、JR with **predict-not-take** policy, is designed as showed in the following figure.



We have finished the pipeline CPU solving structure hazards, data hazards, control hazards and supporting 16 common instructions.

(1) If the following MIPS codes running on the above CPU, does any pipeline stall pop up? If yes, please tell which (structural hazard/ data hazard/ control hazard) causes the stall? Please demonstrate your answer with pipeline status graph of the first iteration executed), and specify where and which forwarding path is used (mark the used forwarding path in the above figure). Assume that r4 does not equal r5 initially.

```
Loop: LW  r4, $0(r1)
      LW  r5, $0(r2)
      BEQ r4, r5, exit
      ADDI r1, r1, #4
      ADDI r2, r2, #4
      J     Loop
exit:
```

(2) Do you know any approach to decrease the stalls for this code segment ? Explain how ?

(3) Now we are required to add the "jalr" instruction to it. The format of "jalr" instruction is

000000	rs	00000	rd	00000	001001

The function of it is rd←PC+4; PC←rs.

Please modify the data path in the given figure to implement the "jalr" instruction and give the values of the control signals in ID stage for "jalr". Please note you might need to add some new control signals.

Introduction to Computer Architecture

Quiz 1 Solution

November 08, 2017

REQUIREMENT:
Refer to only textbooks and lecture slides but NOT sample solutions for previous assignments or quizzes.

1. [5] Analyze why ideal pipelining with equal-length pipe stages yields the highest speedup.

Solution: The analysis should be built on the equation of speedup:
Speedup = (Time per instr when unpipelined) / (Time per instr when pipelined).

Consider, for example, an *n*-stage pipeline, with each stage takes time of t_i. Without considering the overhead by pipeline, we have

Time per instr when unpipelined = $\sum_{i=1}^{n} t_i = T$.

When the pipeline is fully utilized (i.e., components for every stage are processing a different instruction in every clock cycle), the average time for completing an instruction depends on that of the slowest stage.
Time per instr when pipelined =max(t_i).

Speedup = $\sum_{i=1}^{n} t_i$ / max(t_i)

To maximize speedup, we need to find an assignment of the set of t_i such that speed up is maximized while the constraint of $\sum_{i=1}^{n} t_i = T$ is satisfied.

The solution is when $t_i=T/n$ for i = 1…n, that is, ideal pipeline with equal-length stages.

2. [20 = 10+10] Use the following code fragment:

```
Loop:   LD      R1, 0(R2)       ; load R1 from address 0+R2
        DADDI   R1, R1, #1      ; R1 = R1 + 1
        SD      R1, 0(R2)       ; store R1 at address 0+R2
        DADDI   R2, R2, #4      ; R2 = R2 + 4
```

```
DSUB        R4, R3, R2          ; R4 = R3 - R2
BNEZ        R4, Loop            ; branch to Loop if R4 != 0
```

a. Assume that the initial value of R3 is R2 + 396.

Data hazards are caused by data dependences in the code. List all of the data dependences in the code above. Record the register, source instruction, and destination instruction; for example, there is a data dependency for register R1 from the LD to DADDI, that is,

R1 LD DADDI

b. Show the time of this instruction sequence for the 5-stage RISC pipeline with full forwarding and bypassing hardware. Use a pipeline timing chart like that shown in Figure C.5. Assume that the branch is handled by predicting it as not taken. If all memory references take 1 cycle, how many clock cycles does this loop take to execute?

Solution:

a.

R1 LD DADDI
R1 DADDI SD
R2 SD DADDI
R2 DADDI DSUB
R4 DSUB BNEZ
R2 DADDI LD *;consider loop, 2nd DADDI outputs R2 to be used by LD*

b.

Now we are allowed normal bypassing and forwarding circuitry. Branch outcomes and targets are known now at the end of decode.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
LD	R1, 0(R2)	F	D	X	M	W													
DADDI	R1, R1, #1		F	D	s	X	M	W											
SD	R1, 0(R2)			F	s	D	X	M	W										
DADDI	R2, R2, #4					F	D	X	M	W									
DSUB	R4, R3, R2						F	D	X	M	W								
BNEZ	R4, Loop							F	s	D	X	M	W						
(incorrect instruction)										F	s	s	s	s					
LD	R1, 0(R2)										F	D	X	M	W				

Again we have 99 iterations. There are two RAW stalls and a flush after the branch since the branch is taken. The total number of cycles is $9 \times 98 + 12 = 894$. The last loop takes three addition cycles since this latency cannot be overlapped with additional loop instances.

3. [5] What operations do the following instruction indicate?

a. Add R4, 100(R1)

b. Add R1, @(R3)

Solution:
a. Regs[R4] ← Regs[R4] + Mem[100+Regs[R1]];
b. Regs[R1] ← Regs[R1] + Mem[Mem[Regs[R3]]];

4. [10 = 5 x 2] You are designing a system for a real-time application in which specific deadlines must be met. Finishing the computation faster gains nothing. You find that your system can execute the necessary code, in the worst case, twice as fast as necessary.
a. How much energy do you save if you execute at the current speed and turn off the system when the computation is complete?
b. How much energy do you save if you set the voltage and frequency to be half as much?

Solutions:

a. 50%
b. Energy = ½ load × V^2. Changing the frequency does not affect energy–only power. So the new energy is ½ load × (½ $V)^2$, reducing it to about ¼ the old energy.

5. [20 = 5 x 4] For the following assume that values A, B, and C reside in memory. Also assume that instruction operation codes are represented in 8 bits, memory addresses are 64 bits, and register addresses are 6 bits.
For each instruction set architecture shown in Figure A.2 (*stack*, *accumulator*, *register-memory*, *register-register*), how many addresses, or names, appear in each instruction for the code to compute C = A + B, and what is the total code size?

Solution:

1) Stack:
Push A // one address appears in the instruction, code size = 8 bits (opcode) + 64 bits (memory address) = 72 bits;
Push B // one address appears in the instruction, code size = 72 bits;
Add // zero address appears in the instruction, code size = 8 bits;
Pop C // one address appears in the instruction, code size = 72 bits;

Total code size = 72 + 72 + 8 + 72 = 224 bits.

2) Accumulator
Load A // one address appears in the instruction, code size = 8 bits (opcode) + 64 bits (memory address) = 72 bits;
Add B // one address appears in the instruction, code size = 72 bits;
Store C // one address appears in the instruction, code size = 72 bits;

Total code size = 72 + 72 + 8 + 72 = 216 bits.

3) Register-memory

Load R1, A // two addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 64 bits (memory address) = 78 bits;
Add R3, R1, B // three addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 6 bits (register address) + 64 bits (memory address) = 84 bits;
Store R3,C // two addresses appear in the instruction, code size = 78 bits;

Total code size = 78 + 84 + 78 = 240 bits.

4) Register-register

Load R1, A // two addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 64 bits (memory address) = 78 bits;
Load R2, B // two addresses appear in the instruction, code size = 78 bits;
Add R3, R1, R2 // three addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 6 bits (register address) + 6 bits (register address) = 26 bits;
Store R3, C // two addresses appear in the instruction, code size = 78 bits;

Total code size = 78 + 78 + 26 + 78 = 260 bits.

6. [10 = 5 x 2] Reason about PredictedUntaken-PenaltyUntaken0 and PredictedTaken-PenaltyTaken2.

Branch scheme	Penalty unconditional	Penalty untaken	Penalty taken
Flush pipeline	2	3	3
Predicted taken	2	3	2
Predicted untaken	2	0	3

Solution:

The first key point is how to quantify branch penalty.
It corresponds to the number of clock cycles invalidated by branch interruption.
For example, when **PredictedUntaken** strategy is adopted and the branch is indeed **untaken**, the pipeline is not interrupted (as shown in the following diagram) and thus no penalty is involved.

branch instr	IF	ID	EXE	MEM	…
next instr		IF	ID	EXE	…

The other key point is when the branch condition is checked and branch target address is calculated.
Given the **PredictedTaken-PenaltyTaken2** case, based on the following diagram that causes two clock cycles to be invalidated, branch condition and target address should be determined until the end of EXE stage.

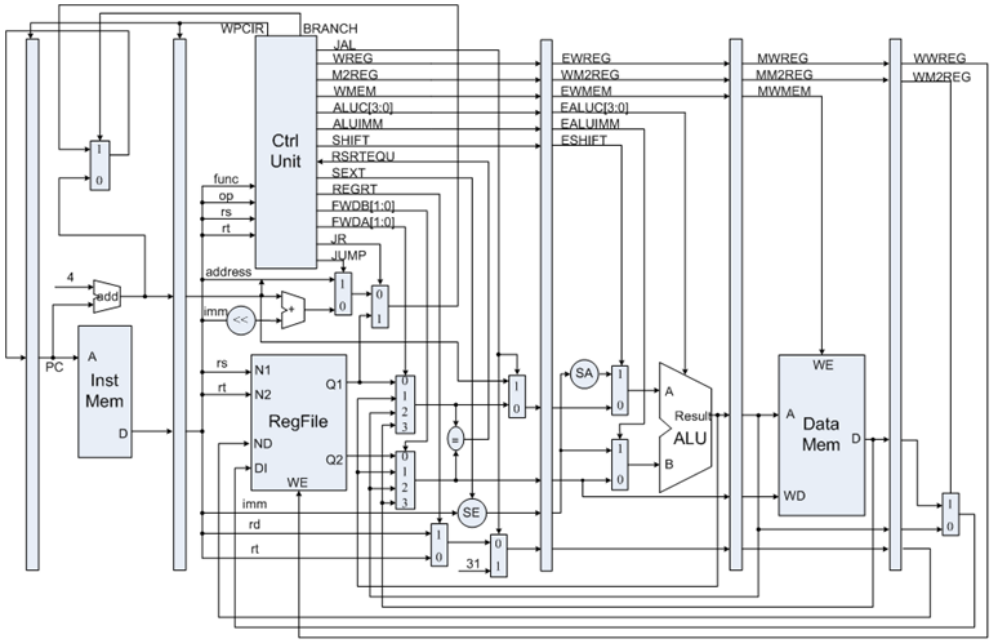
branch instr	IF	ID	EXE	MEM	…
next instr		~~IF~~	~~ID~~		
branch tar				IF	…

7. [20 = 5 + 5 + 10]

a. Which are the five stages of the classic five-stage pipeline for a RISC processor? What operations does each stage perform?

b. Select an example instruction. Mark its datapath on the following figure AND specify which segment corresponds to which pipeline stage.

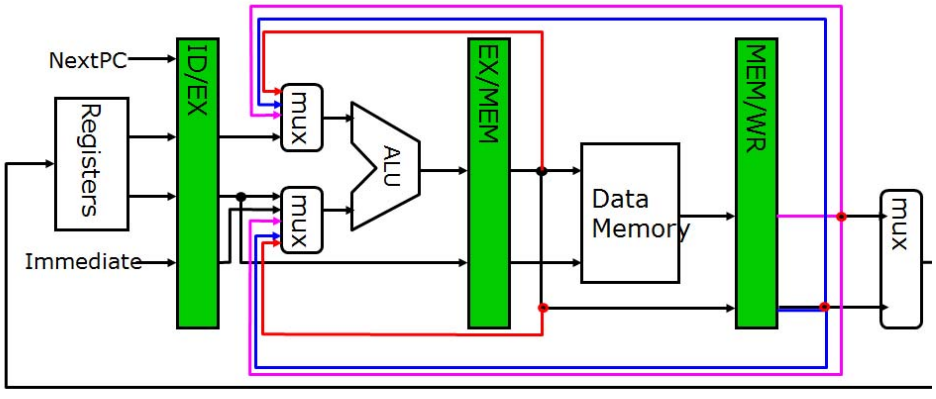
c. How generalized forwarding works against pipeline hazards? Any extra wires are needed for supporting generalized forwarding? If yes, show them on the datapath.



Solution:

c. Generalized forwarding passes a result directly to the functional unit that requires it. It forwards results to not only ALU inputs but also other types of functional units such as memory inputs.

Yes, wires should be added to feed certain outputs back to certain inputs.

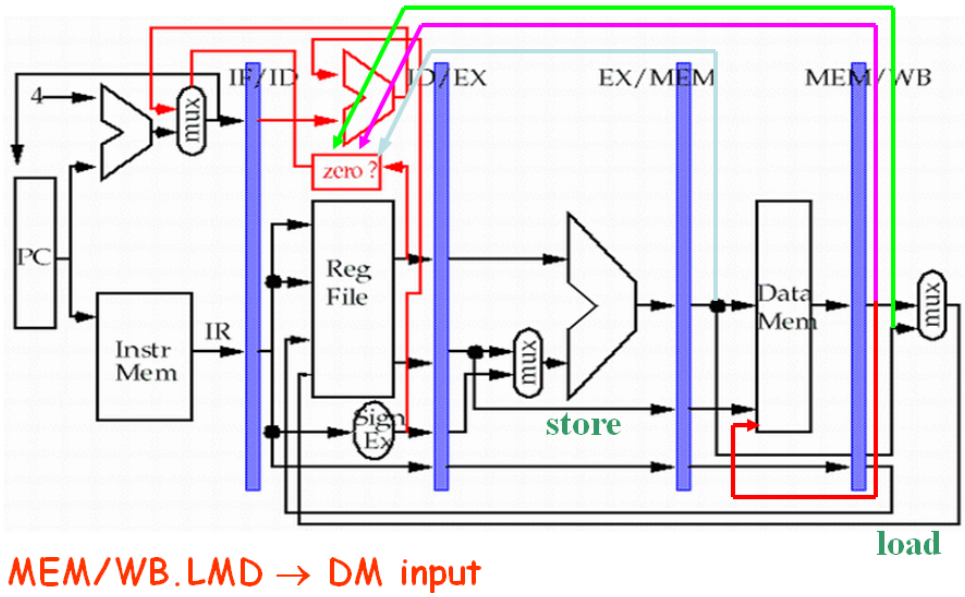


Source → sink

EX/Mem.ALUoutput → ALU input

MEM/WB.ALUoutput → ALU input

MEM/WB.LMD → **ALU** input



8. [10] Design a question that you think is feasible as an exam question.
1) which topic you would like to consider?
2) describe the question;
3) provide also a *correct* sample solution.

Solution:
Sample solution by one dalao:
1) Class
2) Who gives the class candies?
3) Solution: X W T

Introduction to Computer Architecture

Assignment 2

Due January 03, 2018

1. [10 = 8 + 2]
Assume the following RAID 3 system uses even parity and disk2 fails. Determine the original data on disk 2 and fill the form.
Use the first two rows as example to explain how the original data is recovered.

Disk 1	Disk 2 - Failed	Disk 3	Parity
1	1	1	1
0	1	0	1
1	0	1	0
0	0	0	0
0	1	0	1
0	1	0	1
1	0	1	0
1	1	1	1

Even parity requires that the number of bit 1 among original data and parity bit be an even number. Following this rule, the first row needs Disk 2 to be bit 1 and the second row needs Disk 2 to be bit 1, to make the number of bit 1 in the entire row to be even.

The calculation can follow the following equation:
p = sum(data) mod 2,
where p is the bit to be recovered, sum(data) represents the sum of remaining data.
Take the first row for example, sum(data) = 1 + 1 + 1 = 3; 3 mod 2 = 1, which is the recovered data on Disk 2.

2. [10]
Describe the address translation process on the following memory system.

Refer to the textbook, page B-48:

Summary of Virtual Memory and Caches

With virtual memory, TLBs, first-level caches, and second-level caches all mapping portions of the virtual and physical address space, it can get confusing what bits go where. Figure B.25 gives a hypothetical example going from a 64-bit virtual address to a 41-bit physical address with two levels of cache. This L1 cache is virtually indexed, physically tagged since both the cache size and the page size are 8 KB. The L2 cache is 4 MB. The block size for both is 64 bytes.

First, the 64-bit virtual address is logically divided into a virtual page number and page offset. The former is sent to the TLB to be translated into a physical address, and the high bit of the latter is sent to the L1 cache to act as an index. If the TLB match is a hit, then the physical page number is sent to the L1 cache tag to check for a match. If it matches, it's an L1 cache hit. The block offset then selects the word for the processor.

If the L1 cache check results in a miss, the physical address is then used to try the L2 cache. The middle portion of the physical address is used as an index to the 4 MB L2 cache. The resulting L2 cache tag is compared to the upper part of the physical address to check for a match. If it matches, we have an L2 cache hit, and the data are sent to the processor, which uses the block offset to select the desired word. On an L2 miss, the physical address is then used to get the block from memory.

Although this is a simple example, the major difference between this drawing and a real cache is replication. First, there is only one L1 cache. When there are two L1 caches, the top half of the diagram is duplicated. Note that this would lead to two TLBs, which is typical. Hence, one cache and TLB is for instructions, driven from the PC, and one cache and TLB is for data, driven from the effective address.

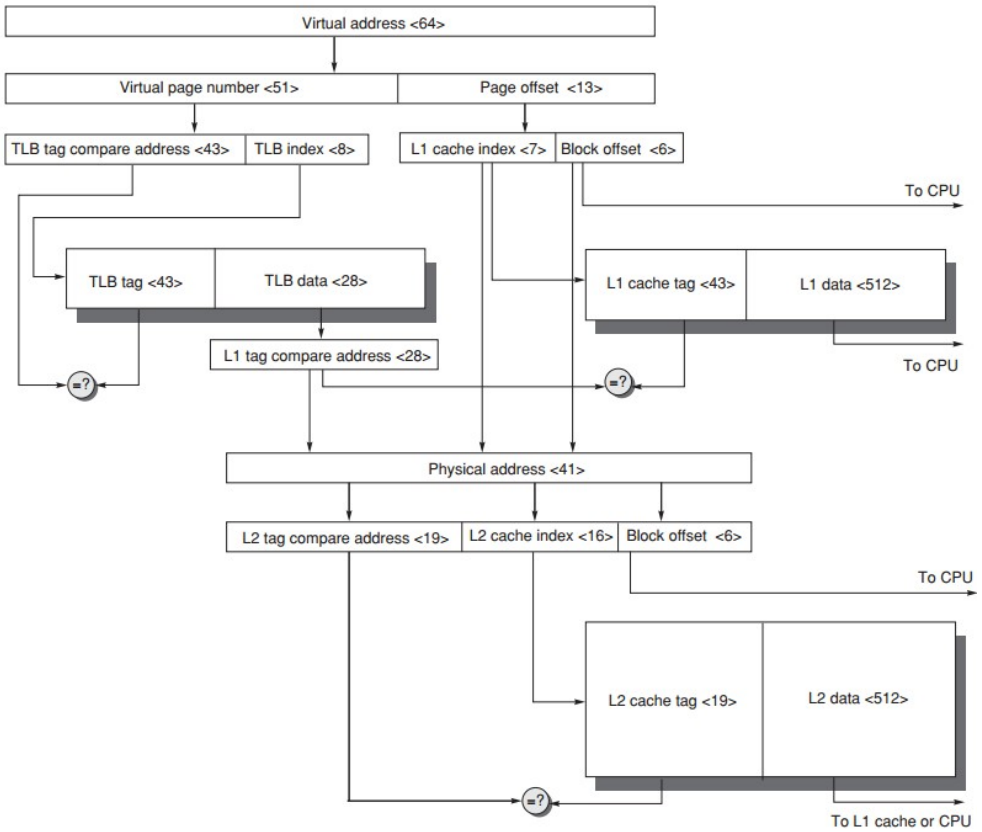


Figure B.25 The overall picture of a hypothetical memory hierarchy going from virtual address to L2 cache access. The page size is 8 KB. The TLB is direct mapped with 256 entries. The L1 cache is a direct-mapped 8 KB, and the L2 cache is a direct-mapped 4 MB. Both use 64-byte blocks. The virtual address is 64 bits and the physical address is 41 bits. The primary difference between this simple figure and a real cache is replication of pieces of this figure.

The second simplification is that all the caches and TLBs are direct mapped. If any were n-way set associative, then we would replicate each set of tag memory, comparators, and data memory n times and connect data memories with an n:1 multiplexor to select a hit. Of course, if the total cache size remained the same, the cache index would also shrink by log2n bits according to the formula in Figure B.7 on page B-22.

3. [10]

Assume a fully associative write-back cache with many cache entries that starts empty. Below is a sequence of five memory operations (the address is in square brackets):

```
Write   Mem[100];
Write   Mem[100];
Read    Mem[200];
Write   Mem[200];
Write   Mem[100];
```

What are the number of hits and misses (and which are them) when using no-write allocate versus write allocate?

Refer to lec08, P49-50

- No-write allocate: 4 misses + 1 hit

cache not affected- address 100 not in the cache;
read [200] miss, block replaced, then write [200] hits;

```
Write Mem[100];  M
WriteMem[100];   M
Read Mem[200];   M
WriteMem[200];   H
WriteMem[100].   M
```

- Write allocate: 2 misses + 3 hits

```
Write Mem[100];  M
WriteMem[100];   H
Read Mem[200];   M
WriteMem[200];   H
WriteMem[100].   H
```

Also refer to textbook, page B-12

For no-write allocate, the address 100 is not in the cache, and there is no allocation on write, so the first two writes will result in misses. Address 200 is also not in the cache, so the read is also a miss. The subsequent write to address 200 is a hit. The last write to 100 is still a miss. The result for no-write allocate is four misses and one hit.

For write allocate, the first accesses to 100 and 200 are misses, and the rest are hits since 100 and 200 are both found in the cache. Thus, the result for write allocate is two misses and three hits.

4. [10]

For the code below, assume we have an 8 KB direct-mapped data cache with 16-byte blocks, and it is a write-back cache that does write allocate. The elements of a and b are 8 bytes long since they a double-precision floating-point arrays. There are 3 rows and 100 columns for a and 101 rows and 3 columns for b. Assume they are not in the cache at the start of the program.

Determine the number of cache misses and which accesses cause them for the following codes with or without prefetching.

```
for (i = 0; i < 3; i = i+1)
        for (j = 0; j < 100; j = j+1)
                a[i][j] = b[j][0] * b[j+1][0];
```

```
for (j = 0; j < 100; j = j+1) {
        prefetch(b[j+7][0]);
        /* b(j,0) for 7 iterations later */
        prefetch(a[0][j+7]);
        /* a(0,j) for 7 iterations later */
        a[0][j] = b[j][0] * b[j+1][0];};
for (i = 1; i < 3; i = i+1)
        for (j = 0; j < 100; j = j+1) {
                prefetch(a[i][j+7]);
                /* a(i,j) for +7 iterations */
                a[i][j] = b[j][0] * b[j+1][0];}
```

Refer to lec10, P38-40

Without prefetching: 251 misses

- **Example: 251 misses**

```
for (i = 0; i < 3; i = i+1)
        for (j = 0; j < 100; j = j+1)
                a[i][j] = b[j][0] * b[j+1][0];
```

16-byte blocks;
8-byte elements for a and b;
write-back strategy;
a[0][0] miss, copy both a[0][0],a[0][1]
as one block contains 16/8 = 2;
so for **a**: 3 x (100/2) = **150 misses**
b[0][0] – b[100][0]: **101 misses**

With prefetching: 19 misses

- **Example: ? misses by prefetching**

```
for (j = 0; j < 100; j = j+1) {
        prefetch(b[j+7][0]);  7 misses: b[0][0] – b[6][0]
        /* b(j,0) for 7 iterations later */
        prefetch(a[0][j+7]);  4 misses: 1/2 of a[0][0] – a[0][6]
        /* a(0,j) for 7 iterations later */
        a[0][j] = b[j][0] * b[j+1][0];};
for (i = 1; i < 3; i = i+1)
        for (j = 0; j < 100; j = j+1) {
                prefetch(a[i][j+7]);  4 misses: a[1][0] – a[1][6]
                                      4 misses: a[2][0] – a[2][6]
                /* a(i,j) for +7 iterations */
                a[i][j] = b[j][0] * b[j+1][0];}
```

- **Example: 19 misses by prefetching**

```
for (j = 0; j < 100; j = j+1) {
        prefetch(b[j+7][0]);  7 misses: b[0][0] – b[6][0]
        /* b(j,0) for 7 iterations later */
        prefetch(a[0][j+7]);  4 misses: 1/2 of a[0][0] – a[0][6]
        /* a(0,j) for 7 iterations later */
        a[0][j] = b[j][0] * b[j+1][0];};
for (i = 1; i < 3; i = i+1)
        for (j = 0; j < 100; j = j+1) {
                prefetch(a[i][j+7]);  4 misses: a[1][0] – a[1][6]
                                      4 misses: a[2][0] – a[2][6]
                /* a(i,j) for +7 iterations */
                a[i][j] = b[j][0] * b[j+1][0];}
```

5. [10]

Look at this code sequence:

SW	R3,	512(R0);	M[512] <- R3	(cache index 0)
LW	R1,	2014(R0);	R1 <- M[1024]	(cache index 0)
LW	R2,	512(R0);	R2 <- M[512]	(cache index 0)

Assume a direct-mapped write-through cache that maps 521 and 1024 to the same block, and a four-word write buffer that is not checked on a read miss. Will the value in R2 always be equal to the value in R3? Explain why.

Refer to textbook, Page B-36

Using the terminology from Chapter 2, this is a read-after-write data hazard in memory. Let's follow a cache access to see the danger. The data in R3 are placed into the write buffer after the store. The following load uses the same cache index and is therefore a miss. The second load instruction tries to put the value in location 512 into register R2; this also results in a miss. If the write buffer hasn't completed writing to location 512 in memory, the read of location 512 will put the old, wrong value into the cache block, and then into R2. Without proper precautions, R3 would not be equal to R2!

6. [10 = 5 + 5]

Suppose that in 1000 memory references there are 40 misses in the first-level cache and 20 misses in the second-level cache. Assume the miss penalty from the L2 cache to memory is 200 clock cycles, the hit time of the L2 cache is 10 clock cycles, the hit

time of L1 is 1 clock cycle, and there are 1.5 memory references per instruction.
a. *What is the average memory access time and*
b. *average stall cycles per instruction?*
Ignore the impact of writes.

Refer to lec08, page 85-87

1. various miss rates?
L1: local = global
40/1000 = 4%
L2:
local: 20/40 = 50%
global: 20/1000 = 2%

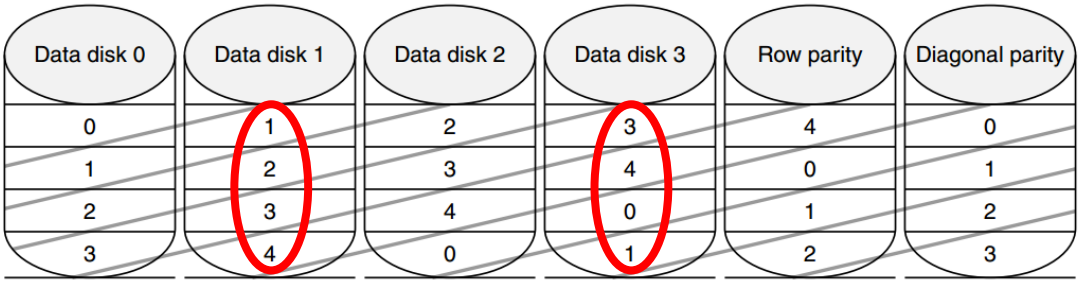
a. average memory access time

2. avg mem access time?
average memory access time
=Hit time$_{L1}$ + Miss rate$_{L1}$
x(Hit time$_{L2}$+Miss rate$_{L2}$xMiss penalty$_{L2}$)
=1 + 4% x (10 + 50% x 200)
=5.4

b. average stall cycles per instruction

3. avg stall cycles per instruction?
average stall cycles per instruction
=Misses per instruction$_{L1}$ x Hit time$_{L2}$
+ Misses per instr$_{L2}$ x Miss penalty$_{L2}$
=(1.5x40/1000)x10+(1.5x20/1000)x200
=6.6

7. [10]
Provide a suitable recovery sequence for the illustrated RAID-DP (or row-diagonal parity) with disks 1 and 3 failed.



Refer to lec11, page 46-60
Multiple recover sequences are possible

8. [10]
To be covered on January 03, please complete this question during class.

Assume that words x1 and x2 are in the same cache block, which is in the shared state in the caches of P1 and P2. Assuming the following sequence of events, identify each miss as a true sharing miss, a false sharing missing, or a hit. Any miss that would occur if the clock size were one word is designated a true sharing miss.

Time	P1	P2
1	Write x1	
2		Read x2
3	Write x1	
4		Write x2
5	Read x2	

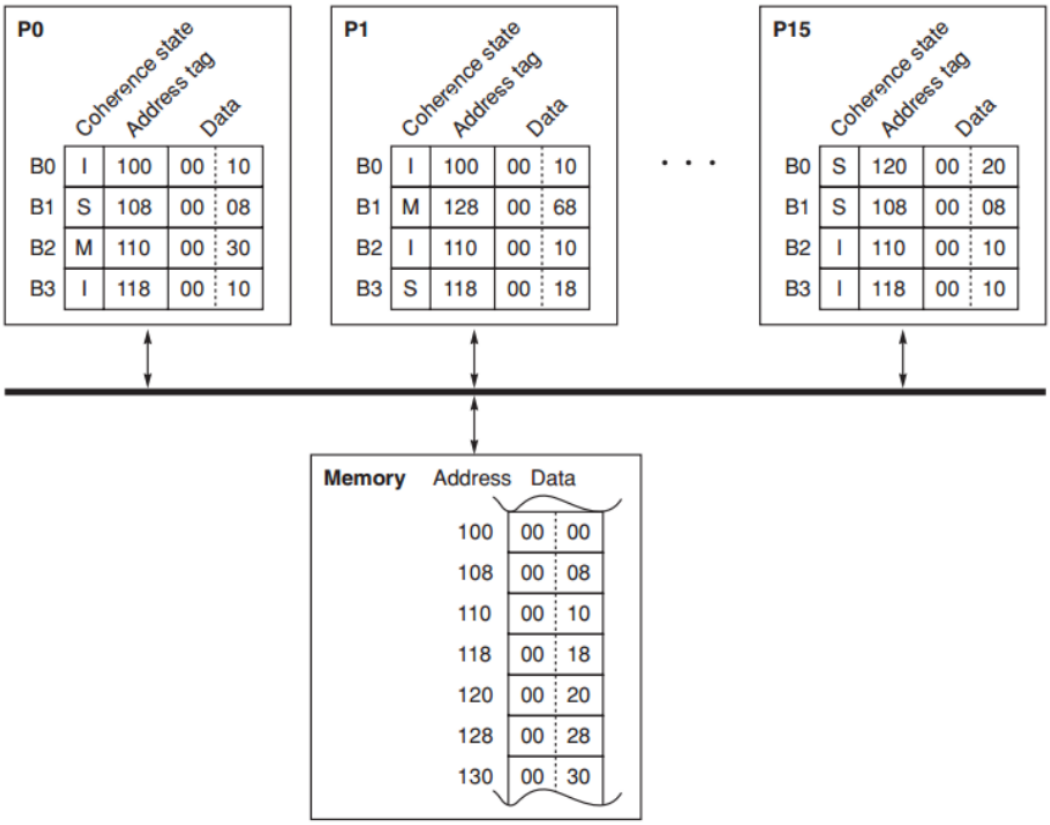
Refer to textbook, P367

Here are the classifications by time step:

1. This event is a true sharing miss, since x1 was read by P2 and needs to be invalidated from P2.
2. This event is a false sharing miss, since x2 was invalidated by the write of x1 in P1, but that value of x1 is not used in P2.
3. This event is a false sharing miss, since the block containing x1 is marked shared due to the read in P2, but P2 did not read x1. The cache block containing x1 will be in the shared state after the read by P2; a write miss is required to obtain exclusive access to the block. In some protocols this will be handled as an *upgrade request*, which generates a bus invalidate, but does not transfer the cache block.
4. This event is a false sharing miss for the same reason as step 3.
5. This event is a true sharing miss, since the value being read was written by P2.

9. [20 = 5 x 4]
To be covered on January 03, please complete this question during class.

The simple, bus-based multiprocessor illustrated in the following figure represents a commonly implemented symmetric shared-memory architecture. Each processor has a single, private cache with coherence maintained using the snooping coherence protocol. Each cache is directed-mapped, with four blocks each holding two words. To simplify the illustration, the cache-address tag contains the full address and each word shows only two hex characters, with the least significant word on the right. The coherence states are denoted M, S, and I for Modified, Shared, and Invalid.



Treat each action below as independently applied to the initial state as given in the figure.

What is the resulting state (i.e., coherence state, tags, and data) of the caches and memory after the given action? Show only the blocks that change, for example, P0.B0: (I, 120, 00 01) indicates that CPU P0's block B0 has the final state of I, tag of 120, and data words 00 and 01.

Also, what value is returned by each read operation?

a. P0: read 120
b. P0: write 120 <-- 80
c. P15: write 120 <-- 80
d. P1: read 110

a. P0: read 120 → P0.B0: (S, 120, 0020) returns 0020

b. P0: write 120 ← 80 → P0.B0: (M, 120, 0080)
P3.B0: (I, 120, 0020)

c. P3: write 120 ← 80 → P3.B0: (M, 120, 0080)

d. P1: read 110 → P1.B2: (S, 110, 0010) returns 0010

Step d changed to "returns 30."